Indonesian Journal of Electrical Engineering and Computer Science
Vol. 33, No. 3, March 2024, pp. 1641~1652
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v33.i3.pp1641-1652

AES-128 reduced-round permutation by replacing the MixColumns function

Jerico S. Baladhay, Edjie M. De Los Reyes
College of Computer Studies, Tarlac State University, Tarlac, Philippines

Article Info

Article history:

Received Oct 24, 2023
Revised Nov 27, 2023
Accepted Dec 25, 2023

Keywords:

Algorithm
Avalanche effect
Encryption
Permutation
Security

ABSTRACT

Ensuring the protection of digital data is of utmost importance in our current reliance on network operations. However, security measures such as data encryption often result in decreased performance speed. This paper enhanced the 128-bit version of the advanced encryption standard (AES) by substituting the MixColumns function with a permutation-based approach and decreasing the overall number of rounds. The evaluation results indicate a substantial enhancement in the speed of encryption and decryption, with a 76.76% improvement in encryption time and a 55.46% improvement in decryption time. Furthermore, it is important to mention that the modifications implemented in the standard AES did not compromise its security in relation to the strict avalanche criterion. The avalanche effect of the modified AES is 52.92%, surpassing the minimum requirement of 50%. Finally, the modified AES demonstrated a 31.12% increase in throughput for encryption and a 25.50% increase for decryption when compared to the original AES, using the sample dataset.

Corresponding Author:

Jerico S. Baladhay
College of Computer Studies, Tarlac State University
Tarlac, Philippines
Email: j.baladhay0785@student.tsu.edu.ph

1. INTRODUCTION

In the era of technological progress, multimedia files have become an indispensable component of our daily existence [1], [2]. These files are generated, distributed, and utilized on a vast magnitude [3], emphasizing the utmost importance of security [4]. The growing dependence on computers and the rising need for effective communication between different locations have presented significant challenges in maintaining the security of digital data [5], [6]. During this progression, the enhanced communication channels resulted in significant challenges concerning privacy [7]. Encrypting digital files has become crucial in safeguarding sensitive content from unauthorized access and ensuring the privacy and integrity of digital assets [8], [9].

Encryption, a fundamental component of cryptography, guarantees data security by converting it into an incomprehensible format [10]. This process employs cryptographic algorithms and keys to encrypt information, rendering it accessible solely with the appropriate decryption key [11]. The advanced encryption standard (AES) algorithm is extensively employed as the standard cryptographic algorithm [12]. The National Institute of Standards and Technology (NIST) introduced the successor to the data encryption standard (DES) in 1997 [13]. The AES encryption algorithm offers different key sizes, namely 128 bits, 192 bits, and 256 bits, which ensure the appropriate level of security for data encryption [14].

Although the AES algorithm is widely recognized for its inherent simplicity [15], there is still room for enhancement [16], [17]. Due to the development of processing and execution concepts that could weaken

the effectiveness of current cryptographic standards, there is an increasing agreement on the need to strengthen and make changes to cryptography [18]. The algorithm comprises four transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey [19].

One of the key challenges lies in the computational complexity of the MixColumns operation in AES encryption [20], which combines multiplication and addition [21]. This complexity imposes a substantial computational burden on the encryption process, leading to slower execution times [22]. To address this issue, a study tackled this challenge where the AES MixColumn function is replaced with a bit permutation to enhance the speed of the encryption process [23]. The approach involves the manipulation of bits within the states, rearranging their order, and altering their values [24], [25]. Therefore, it leads to a reduction in encryption time, lower memory requirement, and manageable implementation. Moreover, several studies have suggested enhancements to the AES algorithm, including revised round keys [26], reduced iterations, and the reduced-round modified AES algorithm (RRMA) [27]. These changes have been effective in increasing processing speed, particularly for smaller files. This paper presents the reduced-round permutation-based AES-128 (RRPBA) by highlighting two modifications in the AES, first is the replacement of the MixColumns transformation to bit permutation technique and second is reducing the number of iteration rounds from ten to six to improve the performance in securing large files in terms of encryption and decryption time and throughput while maintaining an acceptable level of security which is measured in terms of the avalanche effect.

2. RESEARCH METHOD

This section discusses modifications to the AES encryption algorithm. Including the reduction of iteration rounds and replacement of MixColumns with a bit permutation technique and the integration of a precomputed bit permutation process. These changes aim to optimize encryption for larger files.

2.1. Reduced-round permutation-based AES algorithm

There are two modifications to AES: Replacing MixColumns with the bit permutation technique and reducing the iterations of rounds from 10 to 6. The RRMA algorithm approach is adopted [27]. Whereas the algorithm has been modified to reduce the number of round iterations from 10 to 6 as shown Figure 1.

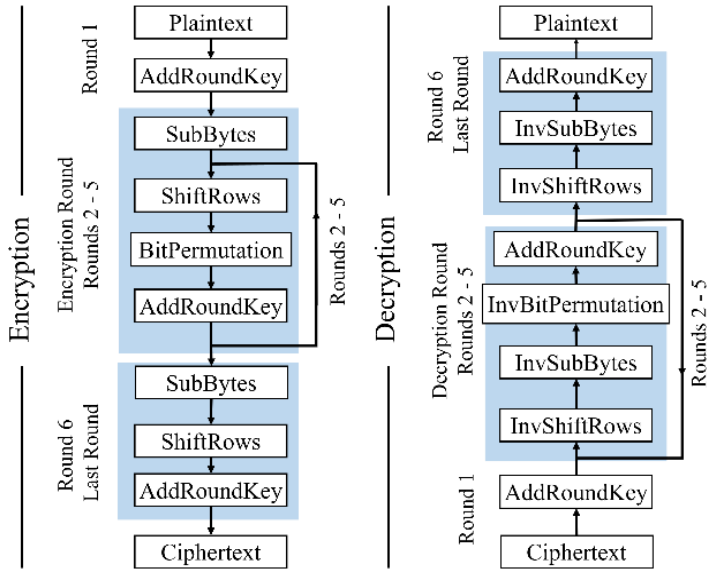

Figure 1. Modified AES encryption and decryption process

The diagram above illustrates the modified encryption process used in the reduced-round permutation-based AES. The process starts with an initial round key, followed by a loop with five encryption rounds. Each round consists of SubBytes, ShiftRows, BitPermutation, and AddRoundKey operation. After completing the five rounds, the process moves to the final sixth round, which includes SubBytes, ShiftRows, and a concluding AddRoundKey operation. Finally, the process ends with the ciphertext, which

signifies the completion of the encryption operation. To decrypt the modified algorithm depicted in Figure 1, the encryption steps are reversed. The process starts with an initial round key and then proceeds to a loop of five decryption rounds. Each round consists of an inverse ShiftRows, inverse SubBytes, inverse BitPermutation, and an AddRoundKey operation. After completing the five rounds, the process moves to the final round, which involves inverse ShiftRows, inverse SubBytes, and an AddRoundKey. Following this process leads to the retrieval of the original plaintext and completes the decryption process.

2.2. Bit permutation

It is based on the bit permutation of DES. The byte data is converted to its binary presentation and is treated as an array. Then implements the bit permutation where the binary presentation is shuffled but instead of the elements, the indices are shuffled. Each element is then mapped to the new position given by these shuffled indices, XORing the result of the bit permutation to the next byte of data to strengthen the encryption, which drastically changes all the results of the algorithm in a flip of a bit. Finally, it converts back the result to the hexadecimal form. To prevent misunderstanding, 128-bit data (i.e., 2b7e151628aed2a6abf7158809cf4f3c) is called a block, the block divided into 4 is called a word (i.e., 2b7e1516, 28aed2a6, abf71588, and 09cf4f3c) and the word divided into 4 is called a byte (i.e., 2b, 7e, 15, and 16).

2.2.1. Permutation table

As illustrated in Figure 2, the permutation table is a set of fixed tables consisting of four distinct index sequences out of 40,320 sequences. The following equation is used to get the total number of possible distinct sequences, $!8 = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 40{,}320$ [28]. The total number of possible index sequences comes from the factorial of 8 since a byte is composed of 8 bits. Each sequence in the table represents the indices of the binary presentation then shifted to the left. It is noteworthy that 'BP' in this context refers to 'bit permutation'.

The purpose of these sequences is to permute every byte in a word, thereby ensuring the safety of transmitted data and preventing unauthorized access. This implies that the bit permutation 1 (BP_1) from the permutation table will be utilized to permute the first byte of a word, and so forth. To obtain the values from the bit permutation table, it starts with the original sequence [0, 1, 2, 3, 4, 5, 6, 7]. The process involves shifting the sequence to the left in each row resulting in a new sequence [1, 2, 3, 4, 5, 6, 7, 0], this is the same process with shift bytes for obtaining the desired values [29].

BP_1	1	2	3	4	5	6	7	0
BP_2	2	3	4	5	6	7	0	1
BP_3	3	4	5	6	7	0	1	2
BP_4	4	5	6	7	0	1	2	3

Figure 2. Permutation table

2.2.2. Inverse permutation table

To inverse the permutation table as illustrated in Figure 3, the following computation is executed to the bit permutation table to get its equivalent inverse permutation table. This will simply reverse the shuffled bits to its original position. Emphasizing the specific meanings of the variables, "n" is representative of a numerical value, "BP" indicates bit permutation, "BPI" denotes bit permutation Index, and "I_BP" represents inverse bit permutation, is crucial. Consider the following steps:

a) Let *n* be a natural number from 0 to 7
b) Let *BP* as the *BP_#* table
c) Let *BPI* as BP_n the value of the *BP* at *index n*
d) *I_BP* be the inverse bit permutation table

Hence, the basic operation is as follows:
For each element from *BP, n=0, 1,,,7* do: $BPI = BP_n$, $I\_BP_{BPI}=n$

I_BP_1	7	0	1	2	3	4	5	6
I_BP_2	6	7	0	1	2	3	4	5
I_BP_3	5	6	7	0	1	2	3	4
I_BP_4	4	5	6	7	0	1	2	3

Figure 3. Inverse permutation table

2.3. Matrix block

The matrix illustrated in Figure 4 represents a block of data. Where in a(r, c): a- represents a byte, r- represents the row index, c-represents the column index. This matrix block [23] is used in computing the result of encryption and decryption of a block of data. This is used in the following algorithms, bit permutation algorithm and inverse bit permutation algorithm.

$a_{(0,0)}$	$a_{(0,1)}$	$a_{(0,2)}$	$a_{(0,3)}$
$a_{(1,0)}$	$a_{(1,1)}$	$a_{(1,2)}$	$a_{(1,3)}$
$a_{(2,0)}$	$a_{(2,1)}$	$a_{(2,2)}$	$a_{(2,3)}$
$a_{(3,0)}$	$a_{(3,1)}$	$a_{(3,2)}$	$a_{(3,3)}$

Figure 4. Block of data a(r, c)

2.4. Bit permutation algorithm

Bit Permutation – is the process of XORing the current byte to the next byte and then shuffling bits of the resulting byte based on bit permutation tables. The process of bit permutation is to shuffle the values of a byte based on the bit permutation table. To understand the process, it is important to know the following, i) arrays - list of data, ii) index - the location or order of the data from the array (represented by a number), and iii) elements - the data in an array. A byte is an array of binary data where the elements can only have 1's and 0's. To permute the byte, the bit permutation table must be used. The data in the table is a sequence of indices where the byte data should be shuffled. Consider a byte that has the following binary data: 10011001. The original index sequence is always: 01234567 where the element in index 0 from the binary data has a value of 1, the element in index 1 has a value of 0, the element in index 2 has a value of 0 and the value of index 3 is 1, and so forth.

To permutate the byte by byte data of each word in a block, follow this process: convert byte data to binary then implement byte permutation as shown in Figure 5, Figure 5(a) which has a bit permutation result 12345670, Figure 5(b) has 23456701, Figure 5(c) 34567012, and Figure 5(d) which result is 45670123, indicating that the bit permutation implementation rearranges elements in the specified sequence, as exemplified in Figures 5(a) to (d), to facilitate a comprehensive word-level analysis of the block. It is crucial to note that the variables "BP", "R", and "FR" hold distinct and specific meanings. The variable "BP" stands for the bit permutation, "R" stands for the result, while "FR" represents the final result. These variables have been assigned specific meanings and cannot be used interchangeably. Let *R1, R2, R3,* and *R4* as the result of XORed bytes. Let *FR1, FR2, FR3,* and *FR4* be the final Bit Permuted results.

a) XOR the *byte(r,0)* to the next byte(r,1), doing the same process to the next bytes except for the last one, where the last result is byte(r,3) XORed to R1.

– Formula:

$R1 = byte(r, 0) \oplus byte(r, 1)$
$R2 = byte(r, 1) \oplus byte(r, 2)$
$R3 = byte(r, 2) \oplus byte(r, 3)$
$R4 = byte(r, 3) \oplus R1$

b) Convert the results to binary.
c) Apply the bit permutation table to the binary.
– Formula:
FR1 = BP_1(R1)
FR2 = BP_2(R2)
FR3 = BP_3(R3)
FR4 = BP_4(R4)
d) Convert back to its hex form.

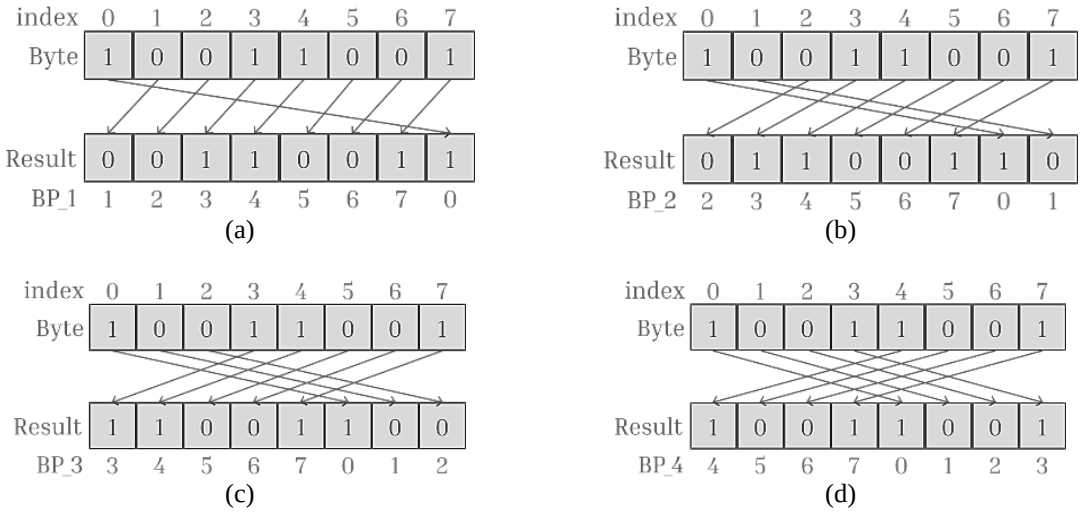

Figure 5. Example of bit permutation implementations: (a) bit permutation 1, (b) bit permutation 2, (c) bit permutation 3, and (d) bit permutation 4

2.5. Inverse bit permutation algorithm

Inverse bit permutation – it is the process of shuffling bits of a byte to their original positions based on inverse bit permutation tables that are equivalent to the bit permutation table used to permute the byte and then XORing it to the next byte or result byte. By computing the word-by-word of the block in a reversed order, one can derive insights into their significance and context. It is of utmost importance to acknowledge that the variables “I_BP”, “R”, and “FR” possess unique and precise connotations. “I_BP” denotes inverse bit permutation, "R" denotes the outcome, and "FR" denotes the ultimate result. Each variable has been designated a specific meaning and cannot be substituted for one another.

To permutate the byte by byte data of each word in a block, follow this process: convert byte data to binary then implement byte permutation as shown in Figure 6, with inverse bit permutation implementation examples using the original sequence 01234567. Figure 6(a) which has an inverse bit permutation result 70123456, Figure 6(b) has 67012345, Figure 6(c) 56701234, and Figure 6(d) which result is 45670123, indicating that the inverse bit permutation implementation restores the original order of bits, as illustrated in Figures 6(a) to 6(d), by inversely applying the bit permutation table to the binary results, emphasizing the crucial role of the given variables stated above in decoding and analyzing the block. Let FR1, FR2, FR3, and FR4 result from XORed bytes. Let R1, R2, R3, and R4 as the final result of bit permuted results:

a) Convert the results to binary.
b) Apply the inverse bit permutation table to the binary.
– Formula:
R4 = I_BP_4(byte(r, 3))
R3 = I_BP_3(byte(r, 2))
R2 = I_BP_2(byte(r, 1))
R1 = I_BP_1(byte(r, 0))
c) XOR R4 to byte(r, 0), for the next bytes, will be XORed with the result of the previous XORed byte.
– Formula:
FR4 = R4 $\oplus$ byte(r, 0)
FR3 = R3 $\oplus$ FR4
FR2 = R2 $\oplus$ FR3
FR1 = R1 $\oplus$ FR2

d) Convert back to its hex form.

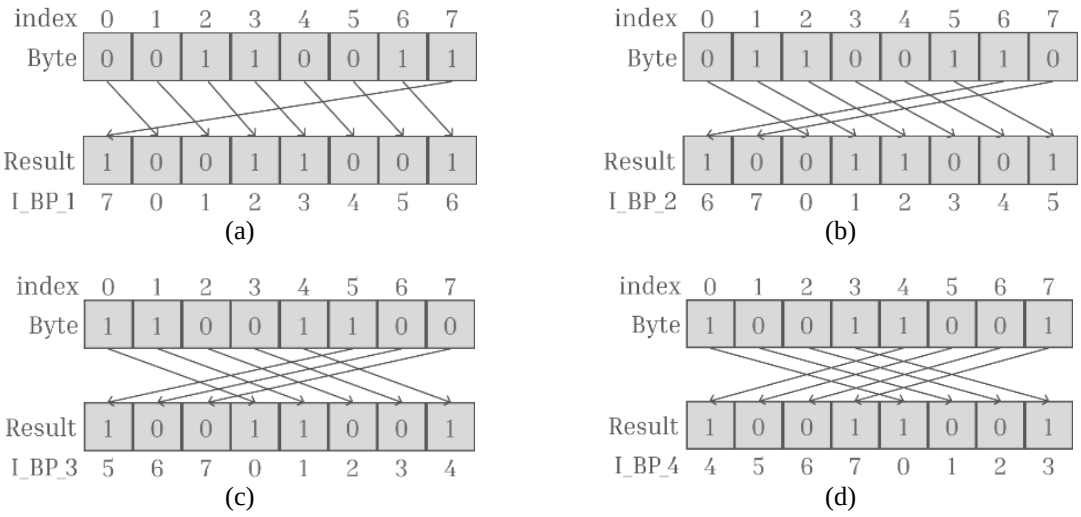

Figure 6. Example of inverse bit permutation implementations: (a) inverse bit permutation 1, (b) inverse bit permutation 2, (c) inverse bit permutation 3, and (d) inverse bit permutation 4

2.6. Precomputed tables

Employing precomputed bit permutation is an effective optimization technique, eliminating the necessity for converting data to binary and performing permutations while keeping the overall process unchanged [30]. To grasp the concept of precomputed tables, consider the multiplication table a prime example. When multiplying two numbers, express 2x5 instead of carrying out the computation 2+2+2+2+2 = 10, using a precomputed table m2[5] = 10, where the table is m2 = [0,2,4,6,8,10]. This approach allows to skip the computation altogether. Precomputed bit permutation skips 2 steps from the process which are the binary conversion and implementation of the bit permutation. Getting the precomputed bit permutation is done using the following:

All possible inputs are from 0 to 255 -> 0 is the lowest in hex and 255 is the max. It's crucial to recognize that the variables "BP", "FR", and "BPF" carry distinct and exact implications. "BP" represents bit permutation, "FR" represents the Final Result, and "BPF" represents the bit permutation function. Each variable has been assigned a specific definition and cannot be interchanged with one another.

a) Let BPF as the bit permutation function that converts numbers to binary and then implements the bit permutation.
b) Let BP as the table for the precomputed bit permutation values.
c) Precompute the bit permutation table BP = [BPF(0), BPF(1),,,,BPF(255)].

To utilize the precomputed bit permutation table, opt for array access rather than employing a function. Instead of FR1 = BP_1(R1), use FR1 = BP_1[R1].

2.7. Metrics

A viable cryptographic algorithm's effectiveness is assessed by its diffusion and confusion properties [31]. Diffusion signifies that even a minor alteration in either the input text or the secret key will significantly change the ciphertext, as gauged by the avalanche effect. Meanwhile, confusion guarantees that the ciphertext provides no insight into the plaintext, and the property is evaluated by examining the randomness of the ciphertext [32]. The cryptographic transformation's effectiveness is assessed by analyzing metrics, time, and throughput. The enumerated provides various tests that were used to evaluate reduced-round permutation-based algorithm:

a) Avalanche effect

– Avalanche effect – assessed the impact of a minor alteration in either the input text or the cipher key on the ciphertext is determined by computing the ratio of the altered bits to the total number of bits present within the ciphertext [33].

Avalanche Effect = Number_of_changed_bit_in_ciphertext / Number_of_bits_in_ciphertext

- Flipping Single Bit – Examining the avalanche effect involved altering a single bit in the plaintext. The aim was to attain a higher percentage with the modified AES compared to the traditional version. Employing the sample plaintext "This letter will change," a single letter was modified to observe the resulting ciphertext. The substantial alteration in the ciphertext value served to gauge the extent of the avalanche effect.

b) Speed test

This section evaluated a set of files through ten (10) encryption and decryption trials. To calculate the average, first find the difference, divide, and multiply it by 100. This process helps to derive a reliable and accurate average result.

- Encryption Duration – The time required for the transformation process is quantified in milliseconds.

The duration of encryption is determined by getting the current timestamp before the encryption starts and then getting the difference between that and the current timestamp after the encryption.

- Decryption Duration – The time required to restore the original transformation process is quantified in milliseconds.

To measure the duration of decryption, it is determined by getting the current timestamp and the current timestamp after the decryption. The method used in modified AES is performance.now() in javascript. This returns the current high-resolution millisecond timestamp, where 0 represents the start of the current node process.

c) Throughput

An evaluation was conducted in this section to analyze the throughput with respect to file size and the average time taken for encryption or decryption. The primary objective of this experiment was to gain insights into the performance of the encryption and decryption processes operating under different file size conditions.

- Encryption Throughput – Termed as the encryption speed, is calculated by multiplying the file size by 1,000 and then dividing it by the encryption time in milliseconds.

Throughput = File size / Encryption (ms).

- Decryption Throughput – The file size and the decryption time determine the computed decryption speed.

Throughput = File size / Encryption (ms)

2.8. Research approach

During the experiments, the secret keys employed are "2b7e151628aed2a6abf7158809cf4f3c" and "2e7e151628aed2a6abf7158809cf4f3c", and text was utilized as indicated in Table 1. The hardware used for the experiment includes a Dual-Core Intel Core i5 CPU operating at 1.8 GHz with 8 GB of RAM. To gauge the avalanche effect and various performance metrics, each test file is encrypted using two distinct cipher keys, and the results are carefully analyzed. The application is built on JavaScript NodeJS, with the CryptoJS library employed for the conventional AES and JavaScript used for the modified AES.

3. RESULTS AND DISCUSSION

Three experiments were conducted: avalanche effect, speed test, and throughput. The research findings show that the modified AES algorithm outperforms the original AES. In terms of performance and strength.

3.1. Sample ciphertext result

This aspect of the study yields valuable insights into the encryption process regarding strength. As shown in Table 1 presents crucial findings highlighting a significant aspect of the modified AES encryption algorithm. Specifically, the analysis demonstrates that the modified AES exhibits an avalanche effect of 52.92%, which surpasses both the expected 50% and the performance of the original AES, which achieved 48.71%, according to the sample dataset. This variance in avalanche effect has implications for data security, implying that the modified AES may provide improved protection against cryptographic attacks and unauthorized access. Consequently, the modified AES represents a more robust encryption option, mainly when data security is a vital concern.

Table 1. Results from sample CipherText of AES vs. modified AES

	KEY 1: 2b7e151628aed2a6abf7158809cf4f3c	KEY 2: 2e7e151628aed2a6abf7158809cf4f3c		KEY 1: 2b7e151628aed2a6abf7158809cf4f3c	KEY 2: 2e7e151628aed2a6abf7158809cf4f3c	
	AES	AES	%	Modified AES	Modified AES	%
1	d8a2e05ebb63e58abd3f37bcffdce9eae7b52f5bf0937474d659a6625b543970	064ac0c8f5260766553e7f54b555b4bc63e760b3579c2f7ef1a9c5df71147f9b	46.09 %	e457055100d5e09374d9ec73c01b2df4ef6044fd0390da3d2aecc90302ef79a8	cae930f12c5d7d19150341081a650605c1cb97632d2647d16d3db9ca8aa1f1a9	50.78 %
2	f60e46d030c350461b5b9b536ffdf1e17f13b11f80b4bc0a91e92197e3e4e6d4	4b9aeda8cdd3acaa6d44b65ad4289c711a1bf7bfd286c0d1edf8df26232966e5	49.60 %	981c4a3717c1edc7e7d32c8e06f26c6e39cda0f211931e4a59c26ca70c2c255a	81e67776d0e0565d2d6fa5477ba9f30520755f3d7c2f15f0cb731d6cd7fe5d8d	57.03 %
3	621d2828f6cc67ebdb6f2056f94e509ed3de409792bb21cb3d1a6be0df7d4180	0497ccc38b5347b10a676cf577c619f9066c034149cb93e862993f0880671226	49.21 %	e12578fb4a2833c59fd3915e807e3063e38d331c0f3740a37887405d33fc61e9	9fb0428c2690c60263cb5adff025fb3b61c7c25598e154fa449a1b42e396800b	51.56 %
4	e2b004d1a0b3951a1ed18ffd909f3a59f816b71f3aea8fe18db275371567ca0c	d295239beb47e77699d43d337a345233dcd65deb1e536bfdffd3fb9ae9b71490	48.43 %	64f829593126ec7a89479965b7e16cedd5badbffa08725c1eb854ce66023d2eb	338640be26d0b3561153a4245dd863a8d38e7d3489ceb92f1220d73765c801f6	52.34 %
5	2c3bb96570f2733ef6831fc7bdbaf6ccd8d2581aff8790af305cbee9f8504a4d	d0b9d69157198e743dc1c7e6104f55ac30081a588e23f55330613009b5abed17	51.17 %	e2eccf909cf146bc6b85bdbeaa283c0a8c16d353816159d0c9a474323ecdfe20	000571442e8504c41cc4311cd22b89f99dbe0875ec86f07f0b584385678b0c9e	53.51 %
6	1defb3d831ddfdfc3cfbbd49f1894d6df5728b5dd1245a4f67585b2b52c1e463	05c86a1ace49f18c35d47db304facd33ad1cf23bbcdd5378415ffa88f0ebddf0	51.17 %	fd34756a15320b82b11a12d0d33cc43e04e61f4d6d47bde0c2c0b2d8e06ce60d	8bca22325c613ed41b618cd8fdab30e27ff8a7d0482f59358d0ee112970534d0	51.56 %
7	5fcbb26ffddd585d594ce4fe6580bddc0c509ad25259f01d906a585409a3c776	46450c899102df22ba3474bb4d8bb7a6d832d6ecefc0448e056a82527f3c8f53	49.21 %	c1d4d48738cf773f5ee7ae6f5933ab9e5fbc753a4991a025cbaccbd887d4bf0a	b1700fa3727adc30885ccb1203b955d38605928e2d240b58e2a58961975201ef	53.12 %
8	61b665345e479d00e3883ccc8f3e20ec0d5ad51ff8fce183ae3b8b9063f51f5b	81c7739e799178e3f4ea4ac0aff40b5712530e977715f0f0ca6bd70172f0707a	46.09 %	45cf7a7220e562cd5516887ccb7f1e6cd13e0050a54344f558d65b05b4fc8b9d	e20bd6ed0cba165a972b6e84dbe5a0852930d2e7e311720dd81eaec677f6f453	52.73 %
9	e72a992676003a3acf04b8c94dcf30ef7a7a490f3e5a705e68a1a2176f9c347d	e7df95c7564a7ee6897ee30a9b78264119e0c4d3284a48f202b5df69fa57d88b	48.82 %	a0e4c0d58d51615022156bdcda68d578a8aa96f8d05af0a7f88ecfff762b7f4f	a83e21684ca24fd3bb685d5b43740cb3fe67c7a888bc4f0b1b0b6cd1a1d1a336	53.90 %
10	7bc2fdfc723e708dfb8f1473979de37e8b32bbef30484853a846760bd6b64bd7	daeb630320a29f2f30acf2321394c807d70c933381531d669b422a3195e6aedb	47.26 %	8d5278a6fd9f33b295f3b3580ce759d6f9ce94fc02cd4aded9a6389d34283670	75fe7ecc87398125a34411660343b3e0aada2f256456dee1242efb5f57bad0c5	52.73 %

3.2. Flipping single bit

In the avalanche effect test as shown in Table 2, a single-bit flip in a different plaintext file revealed that the modified AES algorithm achieved a 53.125% avalanche effect, surpassing the original AES's 46.484375%. This improvement signifies superior bit diffusion capabilities, holding promising implications for bolstering security in various applications. It contributes to advancing cryptographic methods, particularly in ensuring robust encryption and effective propagation of changes in ciphertext, essential for future data protection. In the broader context, this elevated avalanche effect underscores the algorithm's adaptability to evolving security challenges, positioning it as a proactive solution in the ever-changing landscape of cybersecurity. Moreover, the algorithm's consistent performance across various test scenarios reinforces its reliability a suitability for applications demanding robust encryption and resilience against potential attacks.

Table 2. Findings from the avalanche effect test after flipping a single bit in the PlainText

AES	PlainText	Cipher text (Hex)	Avalanche effect
Original AES	This letter will change	cd65ec5dff615c0d795e3d7e41f07f41 9ecae0f2ea73737e50a7172d16fd3e83 fd75e6a5d8d637c5ea82a832d4754e68	0.46484375 (46.484375%)
	This letteq will change	d6c81a6647e80c59646f1098c7e01909	
Secret Key: 2b7e151628aed2a6abf7158809cf4f3c			
Modified AES	This letter will change	14e609315e77c0aeadc8c39a40e9e44a 92cc5cedfd3156f43c80f99b15287617 239454dd043d9d22c77c32a7ec8764fc	0.53125 (53.125%)
	This letteq will change	7a0a6290a64fd036cb793e00586d6213	

3.3. Avalanche effect

As shown in Figure 7 of the avalanche effect graph, the modified algorithm, reduced-round permutation-based algorithm, all of its result is 50% and higher than strict avalanche criterion (SAC) based on the dataset, and the conventional AES, on the other hand, has a lower avalanche result. On the other hand, the conventional AES shows lower avalanche results. The modified algorithm has the potential to serve as a stronger encryption technique, especially in situations where data security is of utmost importance, according to these results. The chart illustrates the percentage results of the ciphertext on the x-axis, with two distinct rows. The upper row showcases outcomes from the modified algorithm, while the lower row displays results from the original algorithm. Meanwhile, the y-axis represents a specific rage of values, determining the position of the line graph, which is contingent on the generated outcome.

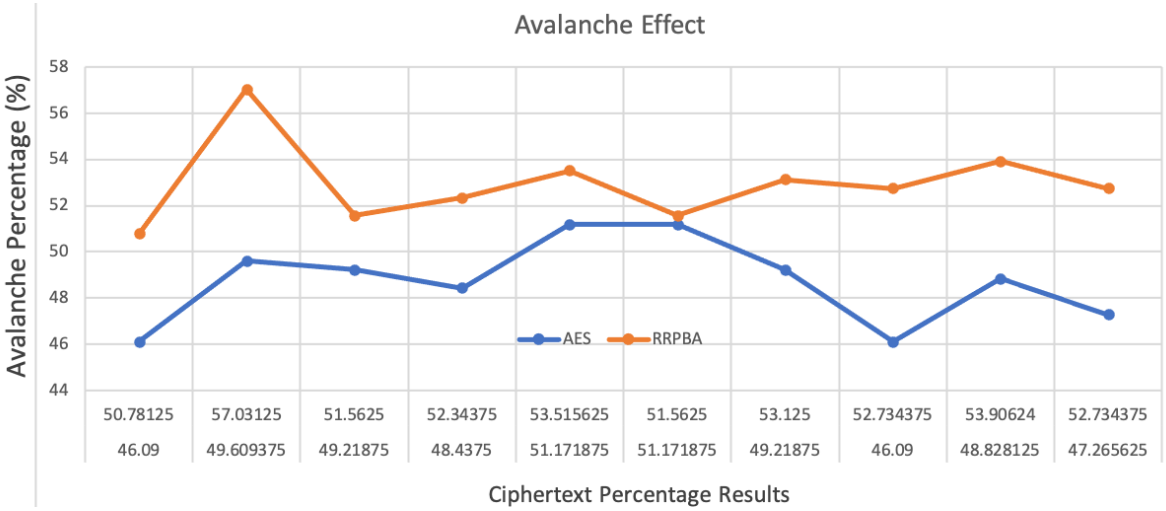

Figure 7. Modified AES and AES avalanche effect graph

3.4. Original AES vs. modified AES time consumption

Comprehensive evaluations, indicate a substantial improvement in the duration of the encryption and decryption process when employing Reduced-Round Permutation-Based Algorithm. Specifically, the modified algorithm demonstrated a remarkable 76.76% improvement in the time of encryption and a significant 55.46% improvement in decryption time, as shown in Table 3. These findings emphasize the speed improvements resulting from the modifications, presenting a noteworthy advantage over the original AES algorithm regarding computational speed. The application of the modified encryption algorithm results in a substantial increase in the time taken for encryption and decryption processes, as shown in comprehensive evaluations. These results highlight the speed improvements brought about by the changes, indicating a significant edge over the original AES algorithm in terms of computational speed. Therefore, the observed acceleration in encrypting and decrypting files of varying sizes highlights the effectiveness of the modified algorithm, showcasing a significant speed advantage over the original AES algorithm.

3.5. Throughput

The data presented in the graph of Figure 8 indicates that the modified algorithm achieves a throughput of 31.12% for encryption and 25.50% for decryption, whereas the original or conventional AES records

7.18% for encryption and 11.32% for decryption. These findings illustrate that the modified algorithm surpasses the original regarding effectiveness and performance. The modified algorithm boasts considerably faster speeds for encryption and decryption, surpassing those of the original AES. This discovery has significant implications for forthcoming applications, especially in situations where computational processes is crucial. The chart depicts two primary operations along the x-axis: encryption and decryption. On the y-axis, each bar signifies numerical values, reflecting the degree of increase or decrease for each respective operation. This visual representation facilitates a clear comparison of throughput metrics between the encryption and decryption processes, providing insight into their respective performance levels.

Table 3. Time consumption comparison: Original AES vs. modified AES

			Encryption time		Decryption time	
File	Size	Resolution	AES (ms)	Modified AES (ms)	AES (ms)	Modified AES (ms)
.MP4	8.8 MB	480p	1154.70	320.91	799.83	351.27
.WMV	15.8 MB	720p	2221.26	499.85	1433.32	567.59
.AVI	25.8 MB	720p	3587.69	767.23	2221.93	1010.06
.MOV	30 MB	1080p	4146.56	935.44	2518.75	1183.43
.MKV	38.2 MB	1080p	5638.35	1236.54	3446.85	1610.51

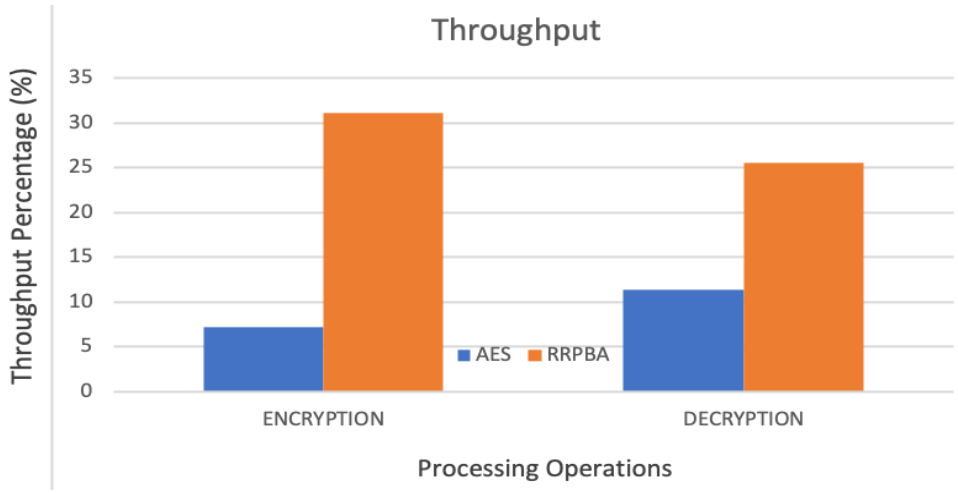

Figure 8. Modified AES and original AES throughput graph

4. CONCLUSION

This paper introduces a modified version of the AES that uses a permutation-based approach with fewer rounds. The purpose of this modification is to encrypt video files. The findings illustrate its efficacy in encrypting and decrypting substantial files. The modification exhibits superior cryptographic strength compared to the original AES, as demonstrated by the avalanche effect results. So far, by reducing the number of rounds and replacing MixColumn with precomputed bit permutation, the applied modifications have enhanced the security in relation to the strict avalanche criterion (SAC). In future development, to optimize the encryption speed and efficiency, the researchers could explore a parallel processing approach that leverages multi-core processors or GPU-based parallelism. This would meet the growing demand for real-time data protection. Additionally, advanced iterations of AES, such as 192 and 256, will be incorporated to support audio and other file formats.

REFERENCES

[1] F. M. Amin and H. Sundari, "EFL students' preferences on digital platforms during emergency remote teaching: Video Conference, LMS, or messenger application?," *Studies in English Language and Education*, vol. 7, no. 2, pp. 362–378, Sep. 2020, doi: 10.24815/siele.v7i2.16929.

[2] M. Napal, A. M. Mendióroz-Lacambra, and A. Peñalva, "Sustainability teaching tools in the digital age," *Sustainability*, vol. 12, no. 8, p. 3366, Apr. 2020, doi: 10.3390/su12083366.

[3] K. Ishii, M. M. Lyons, and S. A. Carr, "Revisiting media richness theory for today and future," *Human Behavior and Emerging Technologies*, vol. 1, no. 2, pp. 124–131, Apr. 2019, doi: 10.1002/hbe2.138.

[4] S. Sharma, T. Kumar, R. Dhaundiyal, A. K. Mishra, N. Duklan, and A. Maithani, "Improved method for image security based on chaotic-shuffle and chaotic-diffusion algorithms," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, p. 273, Feb. 2019, doi: 10.11591/ijece.v9i1.pp273-280.

[5] I. Kuntadi *et al*., "Towards digital TVET: A comparative study on students' readiness in the industry digital demands in Indonesia and Malaysia," *Journal of Technical Education and Training*, vol. 14, no. 3, Dec. 2022, doi: 10.30880/jtet.2022.14.03.008.

[6] C. Katrakazas, A. Theofilatos, G. Papastefanatos, J. Härri, and C. Antoniou, "Cyber security and its impact on CAV safety: Overview, policy needs and challenges," in *Advances in Transport Policy and Planning*, vol. 5, Elsevier, 2020, pp. 73–94, doi: 10.1016/bs.atpp.2020.05.001.

[7] J. Li *et al*., "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 460–474, Jan. 2021, doi: 10.1109/TDSC.2019.2894411.

[8] F. Wiesböck and T. Hess, "Digital innovations: Embedding in organizations," *Electron Mark*, vol. 30, no. 1, pp. 75–86, Mar. 2020, doi: 10.1007/s12525-019-00364-9.

[9] X. Cao and L. Liu, "Use of smart devices: a survey, some research issues, and challenges," in *2020 International Conference on Culture-oriented Science and Technology (ICCST)*, Beijing, China: IEEE, Oct. 2020, pp. 378–382, doi: 10.1109/ICCST50977.2020.00079.

[10] A. M. Qadir and N. Varol, "A review paper on cryptography," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, Barcelos, Portugal: IEEE, Jun. 2019, pp. 1–6, doi: 10.1109/ISDFS.2019.8757514.

[11] M. A. Al-Shabi, "A survey on symmetric and asymmetric cryptography algorithms in information security," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 3, p. p8779, Mar. 2019, doi: 10.29322/IJSRP.9.03.2019.p8779.

[12] K. Muttaqin and J. Rahmadoni, "Analysis and design of file security system advanced encryption standard (AES) cryptography based," *Journal of Applied Engineering and Technological Science (JAETS)*, vol. 1, no. 2, pp. 113–123, May 2020, doi: 10.37385/jaets.v1i2.78.

[13] S. Devi and H. D. Kotha, "AES encryption and decryption standards," *Journal of Physics: Conference Series*, vol. 1228, no. 1, p. 012006, May 2019, doi: 10.1088/1742-6596/1228/1/012006.

[14] U. Arnaut, M. Tair, and M. Veinović, "Comparison of the efficiency of aes implementations on major web platforms," in *Proceedings of the International Scientific Conference - Sinteza 2021*, Beograd, Serbia: Singidunum University, 2021, pp. 153–157, doi: 10.15308/Sinteza-2021-153-157.

[15] M. M. Abu-Faraj and Z. A. Alqadi, "Improving the efficiency and scalability of standard methods for data cryptography," *International Journal of Computer Science and Network Security*, vol. 21, no. 12, pp. 451–458, Dec. 2021, doi: 10.22937/IJCSNS.2021.21.12.61.

[16] J. Kaur, S. Lamba, and P. Saini, "Advanced encryption standard: attacks and current research trends," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India: IEEE, Mar. 2021, pp. 112–116, doi: 10.1109/ICACITE51222.2021.9404716.

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

[18] S. A. Busafi and B. Kumar, "Review and analysis of cryptography techniques," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India: IEEE, Dec. 2020, pp. 323–327, doi: 10.1109/SMART50582.2020.9336792.

[19] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified advanced encryption standard algorithm for information security," *Symmetry*, vol. 11, no. 12, p. 1484, Dec. 2019, doi: 10.3390/sym11121484.

[20] F. Hazzaa, A. M. Shabut, N. H. M. Ali, and M. Cirstea, "Security scheme enhancement for voice over wireless networks," *Journal of Information Security and Applications*, vol. 58, p. 102798, May 2021, doi: 10.1016/j.jisa.2021.102798.

[21] A. Barrera, C.-W. Cheng, and S. Kumar, "Improved mix column computation of cryptographic AES," in *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, South Padre Island, TX, USA: IEEE, Jun. 2019, pp. 229–232, doi: 10.1109/ICDIS.2019.00042.

[22] N. M. S. Surameery, "Modified advanced encryption standard for boost image encryption," *UHD Journal of Science and Technology*, vol. 6, no. 1, pp. 52–59, Apr. 2022, doi: 10.21928/uhdjst.v6n1y2022.pp52-59.

[23] H. V. Gamido, A. M. Sison, and R. P. Medina, "Modified AES for text and image encryption," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 11, no. 3, p. 942, Sep. 2018, doi: 10.11591/ijeecs.v11.i3.pp942-948.

[24] H. V. Gamido, A. M. Sison, and R. P. Medina, "Implementation of modified AES as image encryption scheme," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 6, no. 3, pp. 301–308, Sep. 2018, doi: 10.11591/ijeei.v6i3.490.

[25] H. V. Gamido, "Implementation of a bit permutation-based advanced encryption standard for securing text and image files," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 3, p. 1596, Sep. 2020, doi: 10.11591/ijeecs.v19.i3.pp1596-1601.

[26] E. M. D. L. Reyes, A. M. Sison, and R. Medina, "Modified AES cipher round and key schedule," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 7, no. 1, pp. 29–36, Mar. 2019, doi: 10.11591/ijeei.v7i1.652.

[27] E. M. D. L. Reyes, A. M. Sison, and R. P. Medina, "File encryption based on reduced-round AES with revised round keys and key schedule," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 2, p. 897, Nov. 2019, doi: 10.11591/ijeecs.v16.i2.pp897-905.

[28] A. Roy, S. Wang, B. Meschede-Krasa, J. Breffle, and S. D. Van Hooser, "An early phase of instructive plasticity before the typical onset of sensory experience," *Nature Communications*, vol. 11, no. 1, p. 11, Jan. 2020, doi: 10.1038/s41467-019-13872-1.

[29] O. Hajihassani, S. K. Monfared, S. H. Khasteh, and S. Gorgin, "Fast AES implementation: a high-throughput bitsliced approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2211–2222, Oct. 2019, doi: 10.1109/TPDS.2019.2911278.

[30] N. Franciscus, X. Ren, and B. Stantic, "Precomputing architecture for flexible and efficient big data analytics," *Vietnam Journal of Computer Science*, vol. 5, no. 2, pp. 133–142, May 2018, doi: 10.1007/s40595-018-0109-9.

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

[32] Z. Hu and S. Kais, "A quantum encryption design featuring confusion, diffusion, and mode of operation," *Scientific Reports*, vol. 11, no. 1, p. 23774, Dec. 2021, doi: 10.1038/s41598-021-03241-8.

[33] S. D. Sanap and V. More, "Performance analysis of encryption techniques based on avalanche effect and strict avalanche criterion," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, Coimbatore, India: IEEE, May 2021, pp. 676–679. doi: 10.1109/ICSPC51351.2021.9451784.

BIOGRAPHIES OF AUTHORS

Jerico S. Baladhay is a faculty member serving as an Information Technology College Instructor at Dominican College of Tarlac under the department of College of Computer Studies since 2020. In addition to this, he is also a senior software developer at the same institution under the Management Information Systems Office from 2020. He is pursuing a master's degree in Information Technology at Tarlac State University. The individual's expertise lies in software development, particularly in web applications, mobile applications, and game development. Moreover, he holds skills certifications in completing Cisco Certified Network Associate Version 7 (CCNAv7) courses in Networking. The individual is also a law student at Tarlac State University, where he is actively engaged in the vibrant nexus of legal s tudies and technology within the academic environment of this esteemed institution. His research interests include data security and encryption algorithms. He can be contacted at email: j.baladhay0785@student.tsu.edu.ph.

Dr. Edjie M. De Los Reyes is an esteemed Research Director, Associate Professor IV currently serving at Tarlac State University for 20 years with a strong background in academics. He was designated the university's Research Director and previously served as an Associate Dean from 2014 to 2016. Additionally, he holds numerous skills certifications, including Cisco Certified Network Associate, Cisco Certified Academic Instructor, Microsoft Office Specialist, and Electronic Data Processing Specialist - Programmer. He is an active member of various academic and research organizations, such as the Philippine Society of Information Technology Educators (PSITE), Philippine Schools Universities and Colleges Computer Education and Systems Society (PSUCCESS), and International Association of Multidisciplinary Research. Furthermore, he has published several research papers in Scopus-indexed journals that focus on data security. He can be contacted at email: emdelosreyes@tsu.edu.ph.