Enhancing the Performance of the Advanced Encryption Standard (AES) Algorithm Using Multiple Substitution Boxes

Felicisimo V. Wenceslao, Jr.

Institute of Information and Computer Studies
Northern Iloilo Polytechnic State College, Estancia, Iloilo, Philippines

Abstract: This paper proposes for a modified version of the AES algorithm using multiple substitution boxes (S-Boxes). While many studies have been conducted specifically on modifying the S-Box, these studies were made to replace the Rijndael S-boxes in the AES cipher. We propose to implement two substitution boxes, where the first S-Box is the Rijndael S-box and will be used as is. The second S-Box was constructed through an XOR operation and affine transformation and will replace the MixColumns operation within the internal rounds in the cipher. Based on simulation testing conducted, it was found out that there is a significant difference in the speed performance between the two versions favouring the proposed AES algorithm using multiple S-Box. The findings also revealed that in both encryption and decryption processes, the AES-2SBox performed more efficiently at 27.638% and 108.369% respectively as compared to the original AES algorithm. However, when tested using the avalanche effect, the changes in the output bits were below the minimum expected rate.

Keywords: AES algorithm, S-box**,** Cryptography, Affine Transformation

1. Introduction

In 1997, the National Institute of Standards and Technology (NIST) started a process to identify a replacement for the Data Encryption Standard (DES) which was generally recognized to be not secured due to fast advances in computer processing power. Unlike the selection process for the DES, the Secure Hash Algorithm (SHA-1) and the Digital Signature Algorithm (DSA), NIST had declared that the AES selection process would be open and have invited experts in the field of cryptography and data security from around the world to participate. There were five encryption algorithms that made to the final round of the screening process.

Ultimately, the encryption algorithm proposed by the Belgium cryptographers Joan Daeman and Vincent Rijmen was selected. Prior to selection, Daeman and Rijmen used the name Rijndael (derived from their names) for their algorithm. After adoption, the encryption algorithm was given the name Advanced Encryption Standard (AES) which is in common use today[1].

In 2001, the NIST formally adopted the AES encryption algorithm and published it as a federal standard under the designation FIPS-197. It was chosen because of its security, performance, efficiency, implementability, and low memory requirements.

The AES algorithm relies on a substitution-permutation network and operates quickly in both hardware and software implementations. It uses a round function that is composed of four different byte-oriented transformations namely Substitution Bytes, Shift Rows, Mix Columns and Add Round Key. The Substitution Bytes (S-Box) in AES algorithm plays an important role as it provides confusion in the cipher text. The basic function of S-Box is to transforms the 8 bits input data into 8 bits secret data using a pre-computed look-up-table. On the other hand, the Mix Columns function provides strength against differential and linear attacks due to the complexity of its mathematical operations. These complex mathematical operations may require computational resources in software implementation. We assume that by replacing the Mix Columns function, the speed performance of the AES algorithm will be improved.

It is in this context that this paper aims to propose for a modified AES algorithm using multiple S-Boxes. With the modified version of the AES algorithm, we expect improvement in the speed performance in both the encryption and decryption processes. Hence, we will also compare the AES-Rijndael version and the modified AES algorithm using multiple S-Boxes and evaluate their speed performance properties during encryption and decryption.

2. Review of Related Studies

Modifying the AES cipher has been the subject of numerous studies. Many of these studies were made to changing the original S-Box using some other techniques or proposing new structures.

For instance, [2] proposed for a modified AES algorithm by changing the original structure where the Mix Columns function was replaced with a Permutation function. Hence, the rounds were managed by the IP Table borrowed from the DES algorithm. The results of their investigation showed that their proposed encryption scheme was fast while still provided good security.

In [3], proposed a Key-Dependent S-Box (AES-KDS) to make the AES algorithm stronger. The encryption and decryption process AES-KDS is similar to the original AES cipher as to the number of rounds, data and key size. Each round functions in the modified version resembles that of original AES, but is composed of 5 stages rather than 4 stages. The extra stage named Rotate S-box is introduced at the start of each round function. The other four stages remain the same. However, for the decryption process there are only 4 stages similar in the original cipher. But the InvSubBytes operation is modified to reverse the effect of the Rotate_Sbox operation previously performed in the encryption process. This is followed by a description of key expansion and generation of shift offset-matrix.

In [4], the authors successfully optimized the AES algorithm by proposing a novel method that involved Shift Row and S-Box modifications to map the Mix Column transformation. This approached eliminated the Sub Bytes function. The result of their experiments showed that both encryption and decryption processes an improvement of 86.143% and 13.085% respectively.

In [5], proposed a substitution box that makes use of the RC4 key schedule algorithm (KSA). The resulting matrix is a Key Dependent S-Box based that is dynamically generated based from some key constructed using RC4. The RC4-generated S-Box was used to replace the Rijndael S-Box during the encryption and decryption processes.

In [6], proposed another Key Dependent S-Box aimed at substituting the Rijndael S-Box. In their paper, they modified the AES cipher by placing another phase in the beginning of the round function. They call the extra phase as the S-Box Rotation that rearranges by rotating the original S-Box according to a round key. The round key was derived using the key schedule algorithm. The rotation value is dependent on the entire round key. Their experiments showed that the enhancement on the original AES did not violate the security properties while further introduced confusion without violating the diffusion property.

In [7], proposed to increase the complexity and security of AES S-box by modifying the affine transformation and adding an affine transformation. Performance analysis demonstrated that the improvement in the cryptographic properties of the AES S-box. Moreover, the number of terms in the improved AES S-box algebraic expression was increased. Comparison results suggested that the improved AES S-box has better performance and can readily be applied to AES.

In [8], proposed an enhanced version of the AES-128 algorithm by reducing the number of rounds from 10 rounds to 8 rounds. They assumed that with less number of rounds, it will result in less processing time of the AES algorithm. However, the reduction in the number of rounds to 8 is risky in security attacks such as differential and distinguishing attacks. To compensate such risk, they enhanced the AES algorithm using a hash function to mitigate the attacks. Their proposed AES algorithm, while less in the number of rounds, was equipped with an extra phase called a hash function using the SHA-256 in each round. The results revealed that the hashing function improved the security aspects of the cipher but required more number of operations.

In [9], proposed a modified algorithm of the AES known as the E-AES where the input plaintext and encryption key are mapped into various binary codes instead of giving plaintext and encryption key directly to the AES algorithm. The E-AES algorithm included two more steps to original AES algorithm, which converts plain text into binary with logical and strong XOR operation. Results of the experiments showed that the performance of E-AES is significantly better than AES algorithm. This result is achieved because the cipher produced by E-AES has strong bit level dependence and concluded that E-AES can produce cipher with high avalanche effect in any binary code conversion as compared to AES.

3. Basic Concept of the AES Algorithm

The AES algorithm is a block cipher with a block length of 128 bits. The key which is provided as the input is expanded into an array of key schedule words. Each word has a size of four bytes. The total key schedule for the 128-bit key is 44 words.

AES allows for three different key lengths: 128, 192, and 256 bits. A 128-bit key length will require the cipher module with 10 rounds of substitutions and permutations to encrypt data input (plaintext)[10], 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The decryption process follows the same number of rounds as well. All the computations in AES algorithm are performed on bytes instead of bits. Hence, the 128 bits of plaintext is treated as 16 bytes. These 16 bytes are positioned in a matrix of four rows and four columns. During the encryption and decryption processes, the 16 bytes of data will form a changeable (4*4) array called the state array[11].

For the encryption, the state array consists initially of the input data. This array will keep changing until the final encrypted data is reach. To decrypt the text, the state array start from the encrypted data and will keep changing until the original data is produced. The encryption of AES is carried out in blocks with a fixed block size of 128 bits each. The AES algorithm performs the transformation from plaintext to cipher text and vice versa based on a specified number of repetitions as defined by the key. Figure 1 shows the AES cipher structure.

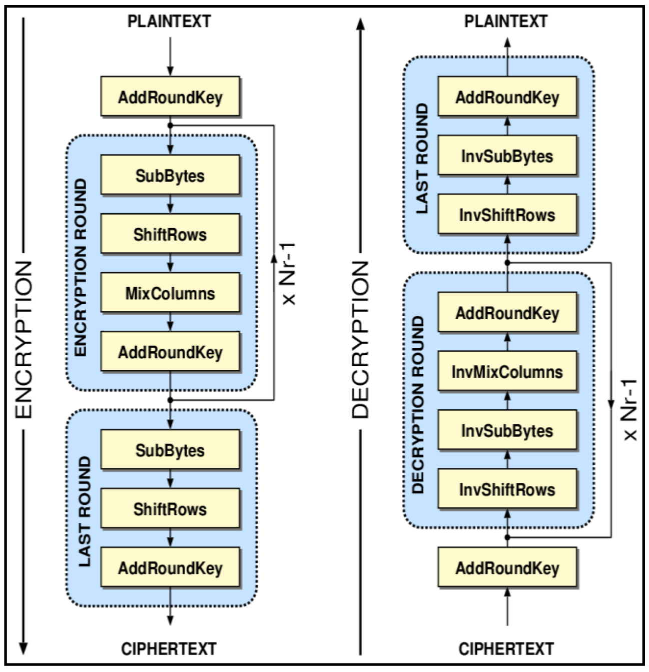

Figure 1. The AES Algorithm Structure.

Except for the last round in each case, all other rounds are identical. Inside each round are four different stages. These are:

3.1 Substitute Bytes (SubBytes) Function

The SubBytes function is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box)[12]. This is a major reason for the security of the AES which is constructed using multiplicative inverse and affine transformation. It provides nonlinearity and confusion. With the help of this lookup table, the 16 bytes of the state are substituted by the corresponding values found in the table. Figure 2 shows the SubBytes operation.

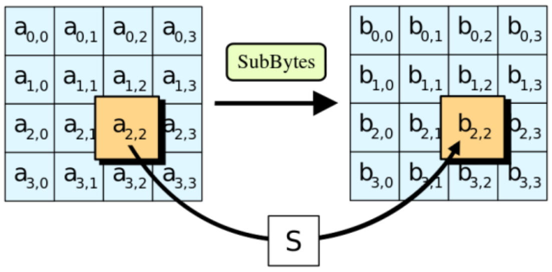

Figure 2. SubBytes Operation

3.2 Shift Rows (ShiftRows Operation)

The ShiftRows operation (figure 3) is a simple byte transposition that provides inter-column diffusion where the bytes in the last three rows of the states are shifted. In this phase, the second row of the state is shifted by one byte position to the left of the matrix; the third row of the state is shifted by two bytes position to the left of the matrix; and the fourth row of the state is shifted by three bytes position to the left[12].

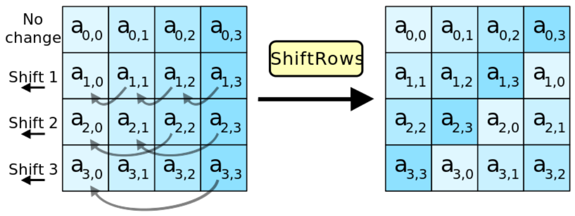

Figure 3. ShiftRows Operation

3.3 Mix Columns (MixColumns Operation)

Figure 4 shows the MixColumns operation that provides to operate on the state column-by-column, treating each column as a four-term polynomial over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial [12]. The bytes are treated as polynomials rather than numbers.

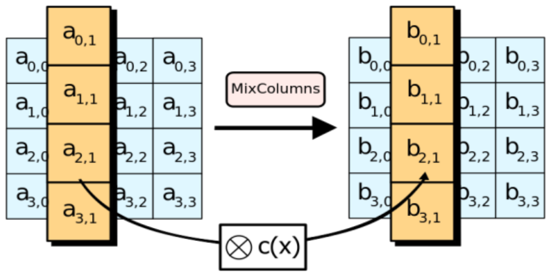

Figure 4. MixColumns Operation

3.4 Add Round Key (AddRoundKey Operation)

The AddRoundKey operation is simple. In this transformation, a round key is added to the State by a simple bitwise XOR operation. Each round key consists of *Nb* words from the key schedule[13]. It is performed by XOR-ing each byte of the State and the round key. Figure 5 shows the AddRoundKey operation.

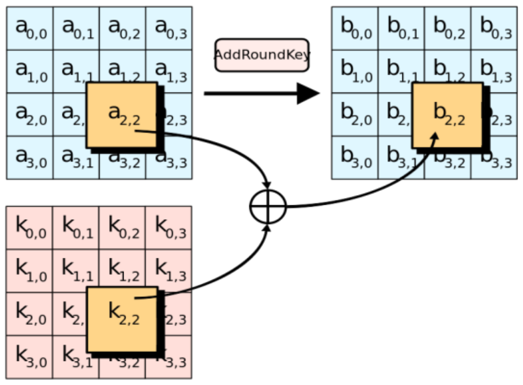

Figure 5. AddRoundKey Operation

For encryption, the individual transformations consist of SubBytes(), ShiftRows(), MixColumns() and AddRoundKey(). These transformations play a role in processing the state[13]. However, the final round is only consists of three stages: SubBytes(), ShiftRows() and AddRoundKey() in producing the final encrypted data or cipher text.
The decryption process is essentially the same structure as the encryption, following the nine rounds of Inverse ShiftRows(), Inverse SubBytes(), Inverse AddRoundKey() and Inverse MixColumns() transformation. In the final round, the Inverse MixColumns() is no longer performed.

4. Proposed AES Algorithm Using Multiple S-Boxes (AES-2Sbox)

Among the four functions within rounds of the AES algorithm, the MixColumns function is perceive to be requiring more computational resources in software implementation as compared to the other functions. This is due to the fact that the MixColumns function provides the critical security properties of the cipher to avoid from linear and/or differential attacks. Conceptually, replacing the MixColumns function by an alternative process may increase the speed performance of the AES algorithm.
In this paper, we propose for a modified version of the 128-bit key length AES algorithm using two substitution boxes. The first S-Box is the Rijndael S-Box that is the default in the original structure of the cipher. It shall be implemented as it is. The second S-Box is constructed using XOR operation and affine transformation. It will replace the MixColumns operations at each round as implemented in the original algorithm.
In essence, the encryption process of the proposed AES-2SBOX algorithm follows the sequence of SubBytes(), ShiftRows(), SubBytesXOR() and AddRoundKey() operations for nine rounds. In the final round, SubBytes(), ShiftRows() and the AddRoundKey() operations will be performed to produce the cipher text.
To decrypt, the proposed AES-2SBOX will transform the ciphertext to plaintext using the sequence of Inverse ShiftRows(), Inverse SubBytes(), Inverse AddRoundKey() and Inverse SubBytesXOR() operations for nine rounds. In the final round, the Inverse SubBytesXOR() is drop to produce the plaintext. Figure 6 shows the proposed modified AES algorithm structure using multiple S-Boxes.

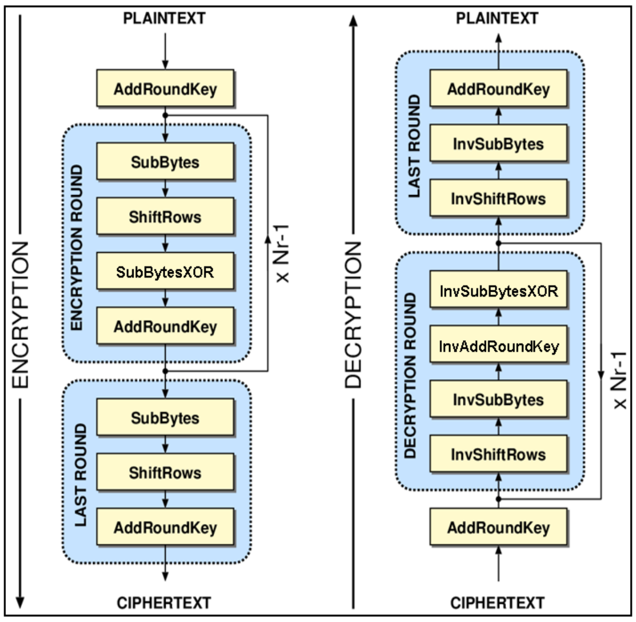

Figure 6. Proposed AES Algorithm Using Multiple S-Boxes

4.1 Construction of the New S-Box

The second S-Box is derived from the original S-Box as designed in the AES (hereafter referred as AES-Rijndael). It is constructed using the following process:

4.1.1 Exclusive OR Operation

The first step is to do an XOR operation to the AES-Rijndael using some Key[i]. The Key[i] shall be any hexadecimal value between 00 to FF. In this particular matrix, the key used was 7F. Hence, the new S-Box shall be referred to as AES-2SBOX$_{7F}$. For the initial values of the AES-SBOX$_{7F}$, each cell in the AES-Rijndael will be XORed with 7F (AES-Rijndael[x,y] ⊗ 7F).

4.1.2 Affine Transform Operation

After creating the initial values of AES-2SBOX$_{7F}$, each cell will be subjected to affine transformation, as applied to the S-Box-Rijndael, to avoid any fix points and to make the new S-box invertible. The affine transformation multiplies each plaintext value by another number and then adds a shift [14]. To scramble the bits in each byte value, we next apply the following transformation to each bit b_i as stored in the initial AES-2SBOX$_{7F}$:

$$b'_i = b_i \otimes b_{(i+4) \bmod 8} \otimes b_{(i+5) \bmod 8} \otimes b_{(i+6) \bmod 8} \otimes b_{(i+7) \bmod 8} \otimes c_i \qquad (1)$$

where c_i is the i^{th} bit of a specially designated byte c whose hex value is 0x63 (c7c6c5c4c3c2c1c0 = 01100011).
For the inverse AES-2SboxXOR, the following transformation to each bit was used for bit scrambling:

$$b'_i = b_{(i+2) \bmod 8} \otimes b_{(i+5) \bmod 8} \otimes b_{(i+7) \bmod 8} \otimes d_i \qquad (2)$$

where d_i is the i^{th} bit of a specially designated byte d whose hex value is 0x05 (d7d6d5d4d3d2d1ddc0 = 00000101).

4.1.3 Matrix Mapping Operation

At the end of the affine transformation, the final values are known. The next step is to map each value to the matrix as appropriate to create the final lookup tables. Table 1 shows the final AES-2SBOX$_{7F}$ while table 2 shows the final Inverse AES-2SBOX$_{7F}$.

Table 1. AES-2SBOX$_{7F}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	13	0C	2D	32	6F	70	51	4E	EB	F4	D5	CA	97	88	A9	B6
1	E2	FD	DC	C3	9E	81	A0	BE	1A	05	24	3B	66	79	58	47
2	F0	EF	CE	D1	8C	93	B2	AD	08	17	36	29	74	6B	4A	55
3	01	1E	3F	20	7D	62	43	5C	F9	E6	C7	D8	85	9A	BB	A4
4	D4	CB	EA	F5	A8	B7	96	89	2C	33	12	0D	50	4F	6E	71
5	25	3A	1B	04	59	46	67	78	DD	C2	E3	FC	A1	BE	9F	80
6	37	28	09	16	4B	54	75	6A	CF	D0	F1	EE	B3	AC	8D	92
7	C6	D9	F8	E7	BA	A5	84	9B	3E	21	00	1F	42	5D	7C	63
8	9C	83	A2	BD	E0	FF	DE	C1	64	7B	5A	45	18	07	26	39
9	6D	72	53	4C	11	0E	2F	30	95	8A	AB	B4	E9	F6	D7	C8
A	7F	60	41	5E	03	1C	3D	22	87	98	B9	A6	FB	E4	C5	DA
B	8E	91	B0	AF	F2	ED	CC	D3	76	69	48	57	0A	15	34	2B
C	5B	44	65	7A	27	38	19	06	A3	BC	9D	82	DF	C0	E1	FE
D	AA	B5	94	8B	D6	C9	E8	F7	52	4D	6C	73	2E	31	10	0F
E	B8	A7	86	99	C4	DB	FA	E5	40	5F	7E	61	3C	23	02	1D
F	49	56	77	68	35	2A	0B	14	B1	AE	8F	90	CD	D2	F3	EC

Table 2. Inverse AES-2SBOX$_{7F}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7A	30	EE	A4	53	19	C7	8D	28	62	BC	F6	01	4B	95	DF
1	DE	94	4A	00	F7	BD	63	29	8C	C6	18	52	A5	EF	31	7B
2	33	79	A7	ED	1A	50	8E	C4	61	2B	F5	BF	48	02	DC	96
3	97	DD	03	49	BE	F4	2A	60	C5	8F	51	1B	EC	A6	78	32
4	E8	A2	7C	36	C1	8B	55	1F	BA	F0	2E	64	93	D9	07	4D
5	4C	06	D8	92	65	2F	F1	BB	1E	54	8A	C0	37	7D	A3	E9
6	A1	EB	35	7F	88	C2	1C	56	F3	B9	67	2D	DA	90	4E	04
7	05	4F	91	DB	2C	66	B8	F2	57	1D	C3	89	7E	34	EA	A0
8	5F	15	CB	81	76	3C	E2	A8	0D	47	99	D3	24	6E	B0	FA
9	FB	B1	6F	25	D2	98	46	0C	A9	E3	3D	77	80	CA	14	5E
A	16	5C	82	C8	3F	75	AB	E1	44	0E	D0	9A	6D	27	F9	B3
B	B2	F8	26	6C	9B	D1	0F	45	E0	AA	74	3E	C9	83	5D	17
C	CD	87	59	13	E4	AE	70	3A	9F	D5	0B	41	B6	FC	22	68
D	69	23	FD	B7	40	0A	D4	9E	3B	71	AF	E5	12	58	86	CC
E	84	CE	10	5A	AD	E7	39	73	D6	9C	42	08	FF	B5	6B	21
F	20	6A	B4	FE	09	43	9D	D7	72	38	E6	AC	5B	11	CF	85

5. Evaluation Results

5.1 Speed Performance

To test the speed performance of the proposed modified AES algorithm using multiple S-Boxes, both versions were used to encrypt and decrypt a file with a size of 50 kilobytes for 50 trials.

For the encryption, the AES-Rijndael version obtained a mean of 175.86ms while the proposed AES-2SBox obtained a mean of 137.78ms. The t-Test for independent samples was used to statistically compute the significant difference in the speed performance. Based from the result of the test, the obtained P-value was 4.44365E-21. This is lower than the 0.05 level of significance, hence there is indeed a significant difference in the speed performance favouring the proposed AES-2SBox version. Table 3 shows the t-Test Statistics for Independent Samples of the encryption process.

Table 3. Speed Performance Between the AES-Rijndael and AES-2SBox During Encryption

t-Test: Two-Sample Assuming Unequal Variances

	AES-Rijndael	*AES-2SBox*
Mean	175.86	137.78
Variance	284.4493878	208.4608163
Observations	50	50
Hypothesized Mean Difference	0	
Df	96	
t Stat	12.12824712	
P(T<=t) one-tail	2.22182E-21	
t Critical one-tail	1.66088144	
P(T<=t) two-tail	4.44365E-21	
t Critical two-tail	1.984984312	

For the decryption process, the AES-Rijndael obtained a computed mean of 191.70ms while the AES-2SBox version obtained yielded a mean of 92.00ms. Similarly, the t-Test for Independent samples was used for statistical computation. Based from the result of the test, the P value obtained was 6.5283E-64 which is lower than the 0.05 level of significance. Hence, there is a significant difference in the speed performance with the proposed AES-2SBox performing better. Table 4 shows the t-Test statistics for independent samples of the decryption process.

Table 4. Speed Performance Between the AES-Rijndael and AES-2SBox During Decryption

t-Test: Two-Sample Assuming Unequal Variances

	AES-Rijndael	*AES-2SBox*
Mean	191.70	92.00
Variance	141.6020408	148.0408163
Observations	50	50
Hypothesized Mean Difference	0	
Df	98	
t Stat	41.42368676	
P(T<=t) one-tail	3.26415E-64	
t Critical one-tail	1.660551217	
P(T<=t) two-tail	6.5283E-64	
t Critical two-tail	1.984467455	

5.2 Performance Efficiency Using the SpeedUp Test Metric

The two versions were compared using the SpeedUp-Test Metric to determine the difference in their performance efficiency. The SpeedUp Test is a metric for relative performance improvement when executing a task. The execution time of a program can be seen as a latency quantity type of the speedup comparison because it is in seconds per program [15].

For latency values, speedup is defined by the following formula:

$$SpeedUP(\%) = \left(\frac{Latency(T_{old})}{Latency(T_{new})} - 1 \right) * 100 \qquad (3)$$

where SpeedUp(%) is the performance efficiency in percent; Latency(T_{old}) is the old mean execution time (i.e., without the improvement) and Latency(T_{new}) is the new mean execution time (i.e., with the improvement)[15][16].
The obtained mean values of the AES-Rijndael and the AES-2SBox during encryption with 175.86ms and 137.78ms respectively. Thus, the performance efficiency showed that the AES-2SBox is 27.638% better than the original version. Similarly, the obtained mean values of the AES-Rijndael and the AES-2SBox during decryption were at 191.70ms and 92.00ms respectively, the performance efficiency also revealed that the AES-2SBox is 108.369% more efficient that the AES-Rijndael version. Table 5 shows the data.

Table 5. Performance Efficiency Using the SpeedUp Test Metric

Algorithm	Mean		SpeedUP (%)
	Latency(T_{old}) (AES-Rijndael)	Latency(T_{new}) (AES-2SBox)	
Encryption	175.86	137.78	27.638%
Decryption	191.70	92.00	108.369%

5.3 Security Performance Through Avalanche Effect

The avalanche effect refers to a desirable property of cryptographic algorithms. The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly with at least half the output bits will be flip[17].
In [18], the avalanche effect can be derived by using equation:

$$Avalanche\ Effect\ (\%) = (NC/TN) * 100 \qquad (4)$$

where *NC* is the number of changed bits in ciphertext and *TN* is the total number of bits in the ciphertext.
Here, we start to calculate the avalanche effect of the AES-2Sbox. The tests were performed by changing the plaintext bit from “11” to “10” and from “FF” to “F0”. The results obtained were 25.000% with 32 bits that were changed and 19.531% with a flip of 25 bits respectively. The following table shows the result of the test for avalanche effect for AES-2Sbox.

Table 5. Avalanche Effects of the AES-2SBox

Plaintext	Ciphertext	Avalanche Effect
11111111111111111111111111111111	FE88B9C6D624C203A4345796445320E4	25.00% (32)
11111111111111111111111111111110	40A3CFC6D624C2D972345796B15320A4	
00112233445566778899AABBCCDDEEFF	9E53C352DBDF8A4F1034CABEAC05B1FB	19.531% (25)
00112233445566778899AABBCCDDEEF0	37F1B112DBDF8A221034848DAC05B1FB	

6. Conclusion

This paper presents a proposed AES algorithm using multiple S-Boxes. The first Sbox (AES-Rijndael) stand as is in the cipher structure. Meanwhile, a new S-Box was constructed using XOR operation and affine transformation. The second S-Box, which we call AES-2SBOX, replaced the MixColumns phase in the AES cipher rounds.

Two set of tests were conducted to compare the original version and the proposed AES-2SBOX algorithm. In the speed performance test, where the two versions, through a software implementation, encrypted and decrypted a file with a size of 50Kb ran for 50 trials. The t-Test statistics set at 0.05 level of significance revealed that in both encryption and decryption, there is a significant difference in the speed performance favouring the proposed AES-2SBox version with the obtained P-values 4.44365E-21 and 6.5283E-64 respectively.

When compared using the SpeedUp Test Metric to determine the difference in their performance efficiency, it was found out that in both encryption and decryption processes, the AES-2SBox performed more efficiently at 27.638% and 108.369% respectively as compared to the original AES algorithm version.

We also performed the test of avalanche effect on the proposed AES-2SBox algorithm. The results of the simulation revealed that the avalanche effect is slightly lower than the minimum expected output of at least 50% bit flip when 1 bit input is altered. The obtained changes in the bit sequence were computed at 25.000% and 19.351% for two set of plaintext.

From these results, we observed that the speed performance greatly increased in the modified AES algorithm using multiple S-Boxes, while the security side has slightly weakened.

References

[1] TownSendSecurity.com. "Introduction to AES Encryption: White Paper", 2016. (online) Available at https://townsendsecurity.com/sites/default/files/AES_Introduction.pdf.

[2] P. Kawle, A. Hiwase, G. Bagde, E. Tekam, R. Kalbande, "Modified Advanced Encryption Standard", International Journal of Soft Computing and Engineering, Vol. 4, No. 1, pp. 21-23, 2014.

[3] G. N. Krishnamurphy, V. Ramaswamy, "Making AES Stronger: AES with Key Dependent S-Box", International Journal of Computer Science and Network Security, Vol. 8, No. 9, pp. 388-398, 2008.

[4] R. Riyaldhi, Rojali and A. Kurniawan, "Improvement of Advanced Encryption Standards Algorithm with Shift Row and S.Box Modification Mapping in Mix Column", 2nd International Conference on Computer Science and Computational Intellegence 2017, 13-14 October 2017, Bali, Indonesia, 2017.

[5] I. Abd-ElGhafar, A. Rohiem, A. Diaa and F. Mohammed, "Generation of AES Key Dependent S-Boxes using RC4 Algorithm", 13th International Conference on Aerospace Sciences & Aviation Technology (ASAT- 13), May 26 – 28, 2009, Military Technical College, Kobry Elkobbah, Cairo, Egypt, 2009.

[6] J. Juremi, R. Mahmod, S. Sulaiman and J. Ramli, "Enhancing Advanced Encryption Standard S-Box Generation Based on Round Key", International Journal of Cyber-Security and Digital Forensics (IJCSDF) Vol. 1, No. 3, pp. 183-188, 2012.

[7] J. Cui, L. Huang, H. Zhong, C. Chang, W. Yang, "An Improved AES S-Box and its Performance Analysis", International Journal of Innovative Computing, Information and Control Vol. 7, No. 5(A), pp. 2291-2302, 2011.

[8] S. Manasa, P. Mullaimalar, G. B. Gnanaprakash Singh and Prof. S. S. Manivannan, "Reducing the Key Generation Time Using Enhanced AES-128 Algorithm to Secure the Data over Wireless Networks", International Journal of Applied Engineering Research Vol. 8, No. 19, pp. 2453-2456, 2013.

[9] A. Singh, "A New Approach to Enhance Avalanche Effect in AES to Improve Computer Security", Journal of Information Technology & Software Engineering, Vol. 5, No. 1, 2015.

[10] S. D. Putra, A. S. Ahmad, S. Sutikno, Y. Kurniawan, "Attacking AES-Masking Encryption Device with Correlation Power Analysis", International Journal of Communication Networks and Information Security (IJCNIS) Vol. 10, No. 2, pp. 397-401, 2018.

[11] V. Pachori, G. Ansari and N. Chaudhary, "Improved Performance of Advance Encryption Standard using Parallel Computing", International Journal of Engineering Research and Applications, Vol. 2, No. 1, pp. 967-971, 2012.

[12] Federal Information Processing Standards Publication 197, Announcing the Advanced Encryption Standard, 2001. (online) Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[13] Man Young Rhee, "Internet Security: Cryptographic Principles, Algorithms and Protocols", John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2003.

[14] D. Bishop, "Introduction to Cryptography with Java Applet", Jones and Barlette Publishers, Inc. Massachusetts, USA. p. 111, 2003.

[15] SpeedUp, 2005. (online) Available at http://en.wikipedia.org/w/index.php?title=Speedup&oldid=648990695

[16] Martin, M., Performance and Benchmarking, Retrieved from: http://www.cis.upenn.edu/ ~milom/cis501-Fall12/lectures/ 04_performance.pdf

[17] Avalanche Effect, 2014. (online) Available at http://en.ikipedia.org/wiki/Avalanche_effect.

[18] G. Patidar, N. Agrawal and S. Tarmakar, "A block based Encryption Model to Improve Avalanche Effect for Data Security", International Journal of Scientific and Research Publications, Vol. 3, No. 1, pp. 1-4, 2013.