Journal of Engineering and Development, Vol. 18, No.2, March 2014, ISSN 1813- 7822

Enhancing The Encryption Process Of Advanced Encryption Standard (AES) By Using Proposed Algorithm To Generate S-Box

Asst. Lect. Hasan M. Azzawi
Department of Electrical Engineering,
University of Technology (U.O.T.), Baghdad, IRAQ
E-Mail: Hasan.azzawi82@uotechnology.edu.iq

Abstract:

Information security is a very important issue in data communication. Any loss or threat to information transfer can prove to be great loss in the process of sending information. Encryption technique plays a main role in information security systems. The advanced encryption standard algorithm is widely accepted due to its strong encryption, complex processing and its resistance to Brute-force attack. This paper presents a proposed algorithm to enhance the encryption process of advanced encryption standard algorithm (AES) by introducing a new algorithm to generate dynamic S-Box from cipher key. This algorithm will lead to generate more secure block ciphers, and solve the problem of the fixed structure S-Boxes and will increase the security level of the AES block cipher system. The main advantage of this algorithm is that many S-Boxes can be generated by changing cipher key. The proposed algorithm leads to increase the complexity of encryption process and makes the differential and linear cryptanalysis more difficult. MATLAB (R2013a) program is used for simulation.

Keywords: **Cryptography, Encryption, Decryption, Rijndael algorithm, Advanced Encryption Standard (AES), Substitution Box (S-Box).**

تحسين عملية التشفير في خوارزمية التشفير المتقدم القياسية باستخدام خوارزمية مقترحة لتوليد (S-Box)

م.م حسن محمود عزاوي
قسم الهندسة الكهربائية/ الجامعة التكنولوجية

الخلاصة :

يعد أمن المعلومات مسألة هامة جدا في نقل البيانات. إن اية خسارة أو تهديدا في نقل المعلومات يمكن أن يكون خسارة كبيرة في عملية ارسال المعلومات. تلعب تقنية التشفير الدور الرئيسي في أنظمة أمن المعلومات. يقدم هذه البحث طريقة مقترحة لتحسين عملية التشفير في خوارزمية التشفير المتقدم القياسية. أن خوارزمية التشفير المتقدم القياسية

الخوارزمية لاقت قبولاً واسعاً بسبب تشفيرها القوي، ومعالجتها المعقده ومقاومتها لعمليات كسر التشفير. يقدم هذا البحث خوارزمية مقترحة لتعزيز عملية التشفير لخوارزمية التشفير المتقدم القياسية (AES) من خلال إدخال خوارزمية جديدة لتوليد (S-Box) ديناميكي من مفتاح التشفير .سوف تؤدي هذه الخوارزمية لتوليد كتلة شفرات أكثر أمنا، وتحل مشكلة بنية الثابتة ل (S-Box) وسوف تزيد من مستوى الأمان من كتلة نظام الشفرات .والميزة الرئيسية لهذه الخوارزمية هو أن العديد من (S-Box) يمكن توليدها عن طريق تغيير مفتاح التشفير. تم تنفيذ الخوارزمية المقترحة على خوارزمية التشفير المتقدم القياسية الاصلية. أن النظام المقترح يزيد من تعقيد عملية التشفير ويجعل تحليل الشفرات التفاضلية الخطية أكثر صعوبة. تم أستخدام برنامج MATLAB (R2013a) في عملية المحاكاة.

1. Introduction

Transmission of important data over the communication channel have emphasized the need for fast and secure digital communication networks to achieve the requirements for secrecy, integrity and nonproduction of exchanged information. Cryptography provides a method for securing and authenticating the transmission of information over insecure channels. It enables us to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot read it.

The urgency for secure exchange of digital data resulted in large quantities of different encryption algorithms which are evaluated on the basis of throughput, speed of operation and area requirements. There are mainly two types of cryptographic algorithms:

1- Symmetric.
2- Asymmetric algorithms.

Symmetric systems such as Data Encryption Standard(DES), 3DES and Advanced Encryption Standard(AES) uses an identical key for the sender and receiver; both to encrypt the message text and decrypt the cipher text. Asymmetric systems such as Rivest-Shamir-Adelman (RSA) & Elliptic Curve Cryptosystem (ECC) uses different keys for encryption and decryption. Symmetric cryptosystems is more suitable to encrypt large amount of data with high speed. To replace the old Data Encryption Standard, in September 12 of 1997, the National Institute of Standard and Technology (NIST) required proposals to what was called Advanced Encryption Standard (AES) [1]. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called (cipher-text). Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128, 192, or 256 bits [2]. Block cipher systems (e.g. AES) depend on the S-boxes, which are fixed and have no relation with the secret key. The only changeable parameter is the secret key. Since the only nonlinear component of AES is S-boxes, they are an important source of cryptographic strength. The use of key-dependent S-boxes in block cipher design has not been widely investigated in the literature. Research into S-box design has focused on determination of S-box properties which yield cryptographically strong ciphers, with the aim of selecting a small number of good S-boxes for use in a block cipher. Some results have demonstrated that a randomly chosen S-box of sufficient size will have several of these desirable properties with high probability [3]. In this paper a proposed method for constructing cryptographically strong variable S-box dependent key is introduced.

.

2. AES Algorithm

The AES algorithm specifies three encryption modes: 128-bit, 192- bit, and 256-bit. Each cipher mode has a corresponding number of rounds N_r based on key length of N_k words. The state block size, termed N_b, is constant for all encryption modes. This 128-bit block is termed the state. Each state is comprised of 4 words. A word is subsequently defined as 4 bytes. Both encryption and decryption begin with the round key expansion created by the key schedule function [4]. **Table (1)** shows the possible key/state block/round combinations [5].

Table (1) AES Categorization [5].

KeySize (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Number of Rounds	10	12	14
Expanded key size (words/byte)	44/176	52/208	60/240

2.1 Principle of AES Algorithm

The AES encryption and decryption procedures are shown in figures (1) and (2). After an initial round key addition, a round function consisting of four different transformations SubByte(), ShiftRow(), MixColumn(), and AddRoundKey() are applied to the data block (i.e., the state array). The round function is performed iteratively 10, 12, or 14 times, depending on the key length. Note that in the last round MixColumn() is not applied. The four transformations are described briefly as follows [6,7]:

1. SubByte(): a nonlinear byte substitution that operates independently on each byte of the state using a substitution table (the S-Box).

2. ShiftRow(): a circular shifting operation on the rows of the state with different numbers of bytes (offsets).
3. MixColumn(): the operation that mixes the bytes in each column by the multiplication of the state with a fixed polynomial modulo $x^4 + 1$.

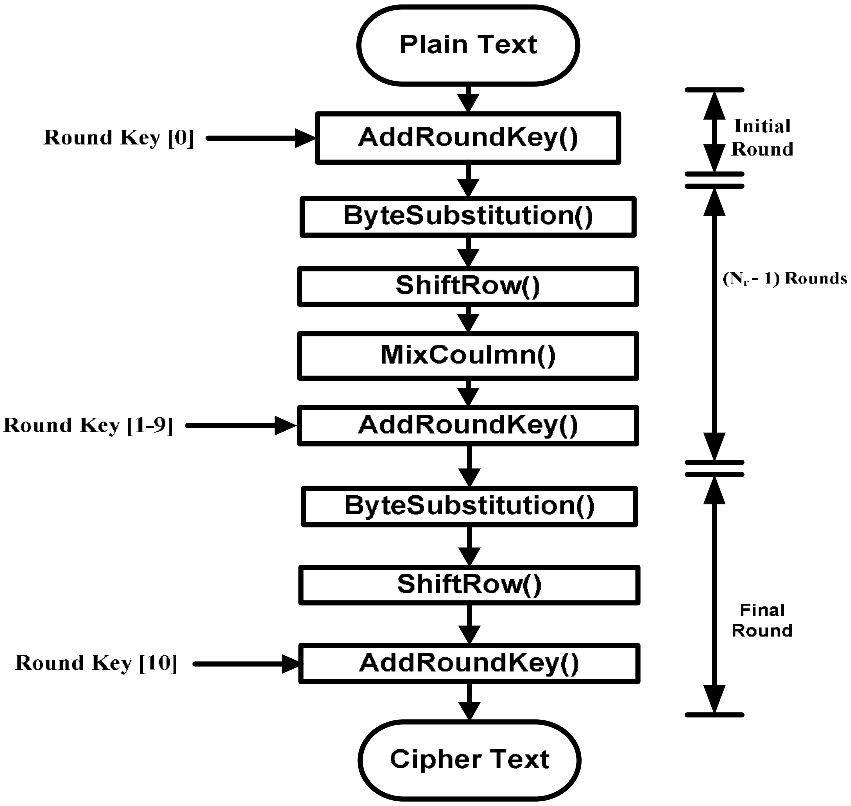


Fig .(1) The Encryption procedure of AES Algorithm [7].

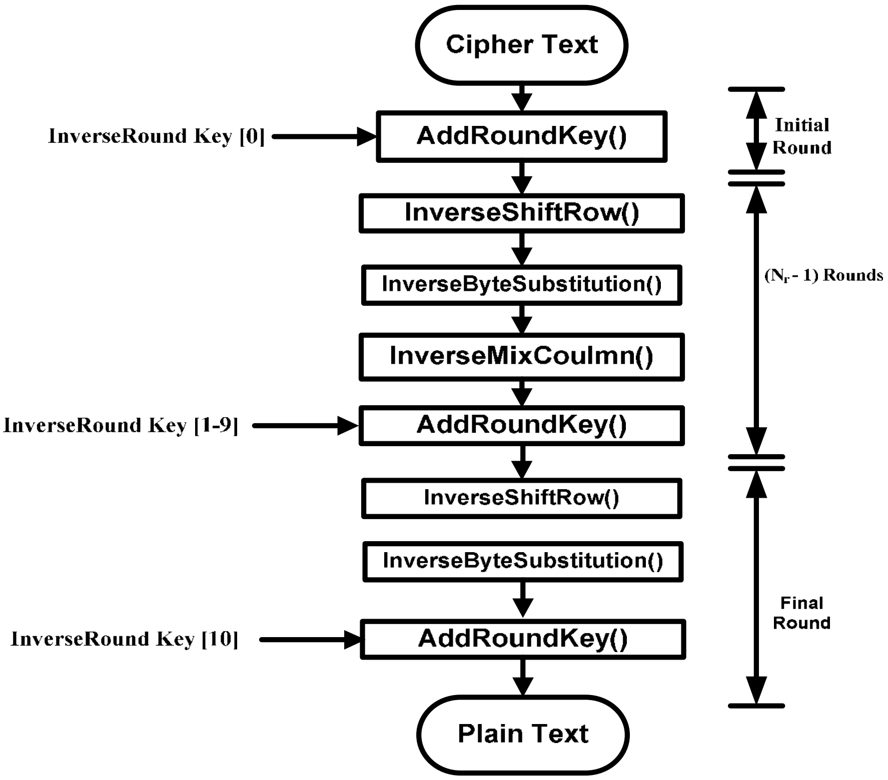


Fig .(2) The Decryption procedure of AES Algorithm [7].

4. AddRoundKey(): an XOR operation that adds a round key to the state in each iteration, where the round keys are generated during the key expansion phase.
5. The Key expansion generates a total of N_b (N_r + 1) words: the algorithm requires an initial set of N_b words, and each of the N_r rounds requires N_b words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [w_i], with *i* in the range $0 < i < N_b (N_r + 1)$.

The decryption of the data which was encrypted using the AES is done by inverting all the encryption operations with the same key with which it is encrypted since the AES is a symmetric encryption standard. In the decryption process the sequence of the transformations differs from that of the encryption but the key expansion for encryption and decryption are the same. However several properties of the AES algorithm allow for an equivalent decryption with the same sequence of transformations as that in encryption.
The operations of the decryption are described briefly as follows [8]:

1. Inverse Sub Bytes: This operation is same as it is in the encryption process but the only difference is the inverse of the substitution box is used here since the substitution box which we used in the encryption is invertible.
2. Inverse Shift Rows: The inverse shift rows operation inverses the shift row operation in the encryption process by right shifting the elements in the rows.

3. Inverse Add Round Key: The add round key process is the same as that of the one in the encryption process.
4. Inverse Mix Columns: In inverse mix column operation the same operation in the mix column is done but with the different matrix as specified below.

The decryption procedure is shown in **Figure (2).**

2.2 S-Box Calculations

In this paper, a new design for enhancing the security of AES algorithm is proposed. This approach design will not contradict the security of the original AES algorithm by keeping all the mathematical criteria of AES remain unchanged. We try to improve the security of AES by making its S-box to be key-dependent.

S-Box is one of the most crucial keystones that will lead to the security at AES level as it is nonlinear, invertible transformation. S-box entries are computed using the multiplicative inverses in Galois Field GF (2^8). This multiplicative inverse uses the affine mapping concept for encryption part and inverse affine mapping concept for decryption part [5]. **Figure (3)** and **Figure (4)** shows the S-box and inverse S-box with 256 8-bit values. Each individual byte of state is mapped into a new byte in the following way: The leftmost 4 bits are used as a row value and the rightmost 4 bits are

63 7c 77 7b f2 6b 6f c5 30 01 67 2b fe d7 ab 76
ca 82 c9 7d fa 59 47 f0 ad d4 a2 af 9c a4 72 c0
b7 fd 93 26 36 3f f7 cc 34 a5 e5 f1 71 d8 31 15
04 c7 23 c3 18 96 05 9a 07 12 80 e2 eb 27 b2 75
09 83 2c 1a 1b 6e 5a a0 52 3b d6 b3 29 e3 2f 84
53 d1 00 ed 20 fc b1 5b 6a cb be 39 4a 4c 58 cf
d0 ef aa fb 43 4d 33 85 45 f9 02 7f 50 3c 9f a8
51 a3 40 8f 92 9d 38 f5 bc b6 da 21 10 ff f3 d2
cd 0c 13 ec 5f 97 44 17 c4 a7 7e 3d 64 5d 19 73
60 81 4f dc 22 2a 90 88 46 ee b8 14 de 5e 0b db
e0 32 3a 0a 49 06 24 5c c2 d3 ac 62 91 95 e4 79
e7 c8 37 6d 8d d5 4e a9 6c 56 f4 ea 65 7a ae 08
ba 78 25 2e 1c a6 b4 c6 e8 dd 74 1f 4b bd 8b 8a
70 3e b5 66 48 03 f6 0e 61 35 57 b9 86 c1 1d 9e
e1 f8 98 11 69 d9 8e 94 9b 1e 87 e9 ce 55 28 df
8c a1 89 0d bf e6 42 68 41 99 2d 0f b0 54 bb 16

Fig .(3) S-Box [14,15].

52 09 6a d5 30 36 a5 38 bf 40 a3 9e 81 f3 d7 fb
7c e3 39 82 9b 2f ff 87 34 8e 43 44 c4 de e9 cb
54 7b 94 32 a6 c2 23 3d ee 4c 95 0b 42 fa c3 4e
08 2e a1 66 28 d9 24 b2 76 5b a2 49 6d 8b d1 25
72 f8 f6 64 86 68 98 16 d4 a4 5c cc 5d 65 b6 92
6c 70 48 50 fd ed b9 da 5e 15 46 57 a7 8d 9d 84
90 d8 ab 00 8c bc d3 0a f7 e4 58 05 b8 b3 45 06
d0 2c 1e 8f ca 3f 0f 02 c1 af bd 03 01 13 8a 6b
3a 91 11 41 4f 67 dc ea 97 f2 cf ce f0 b4 e6 73
96 ac 74 22 e7 ad 35 85 e2 f9 37 e8 1c 75 df 6e
47 f1 1a 71 1d 29 c5 89 6f b7 62 0e aa 18 be 1b
fc 56 3e 4b c6 d2 79 20 9a db c0 fe 78 cd 5a f4
1f dd a8 33 88 07 c7 31 b1 12 10 59 27 80 ec 5f
60 51 7f a9 19 b5 4a 0d 2d e5 7a 9f 93 c9 9c ef
a0 e0 3b 4d ae 2a f5 b0 c8 eb bb 3c 83 53 99 61
17 2b 04 7e ba 77 d6 26 e1 69 14 63 55 21 0c 7d

Fig .(4) Inverse S-Box [14,15].

used as a column value. These row and column values serve as indices into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2a}: The S-box is constructed in the following points [9]:

- Initialize the S-box with the byte values in ascending order row by row. Thus, the value of the byte at row x, column y is {xy}.
- Map each byte in the S-box to its multiplicative inverse in the finite field GF (2^8) the value {00} is mapped to itself.
- Consider that each byte in the S-box consists of 8 bits labeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). The general affine transformation to each bit of each byte in the S-box is:

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \quad \ldots\ldots\ldots\ldots\ldots(1)$$

Where: c_i is the i-th bit of byte c with the value {63}, that is, (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0) = (01100011).

The prime (') indicates that the variable is to be updated by the value on the right. The AES standard depicts this transformation in matrix form as follows:

Each element in the product matrix is the bitwise XOR of elements of one row and one column. Further, the final addition, as shown in equations (2 - 9), is a bitwise XOR, the inverse S-box is obtained by taking the inverse of equation (10), affine transformation followed by taking the multiplicative inverse in GF(2^8) given in figure (4). As an example, consider the input value {95}. The multiplicative inverse in GF (2^8) is $\{95\}^{-1}$ = {8A}, which is 10001010 in binary. Using equation (10), the result is {2A}. More details of affine transformation and S-box are available in the literature [10-13].

$$b_0' = b_0 \oplus b_{(0+4)\bmod 8} \oplus b_{(0+5)\bmod 8} \oplus b_{(0+6)\bmod 8} \oplus b_{(0+7)\bmod 8} \oplus c_0 \quad \text{(2)}$$

$$b_1' = b_1 \oplus b_{(1+4)\bmod 8} \oplus b_{(1+5)\bmod 8} \oplus b_{(1+6)\bmod 8} \oplus b_{(1+7)\bmod 8} \oplus c_1 \quad \text{(3)}$$

$$b_2' = b_2 \oplus b_{(2+4)\bmod 8} \oplus b_{(2+5)\bmod 8} \oplus b_{(2+6)\bmod 8} \oplus b_{(2+7)\bmod 8} \oplus c_2 \quad \text{(4)}$$

$$b_3' = b_3 \oplus b_{(3+4)\bmod 8} \oplus b_{(3+5)\bmod 8} \oplus b_{(3+6)\bmod 8} \oplus b_{(3+7)\bmod 8} \oplus c_3 \quad \text{(5)}$$

$$b_4' = b_4 \oplus b_{(4+4)\bmod 8} \oplus b_{(4+5)\bmod 8} \oplus b_{(4+6)\bmod 8} \oplus b_{(4+7)\bmod 8} \oplus c_4 \quad \text{(6)}$$

$$b_5' = b_5 \oplus b_{(5+4)\bmod 8} \oplus b_{(5+5)\bmod 8} \oplus b_{(5+6)\bmod 8} \oplus b_{(5+7)\bmod 8} \oplus c_5 \quad \text{(7)}$$

$$b_6' = b_6 \oplus b_{(6+4)\bmod 8} \oplus b_{(6+5)\bmod 8} \oplus b_{(6+6)\bmod 8} \oplus b_{(6+7)\bmod 8} \oplus c_6 \quad \text{(8)}$$

$$b_7' = b_7 \oplus b_{(7+4)\bmod 8} \oplus b_{(7+5)\bmod 8} \oplus b_{(7+6)\bmod 8} \oplus b_{(7+7)\bmod 8} \oplus c_7 \quad \text{(9)}$$

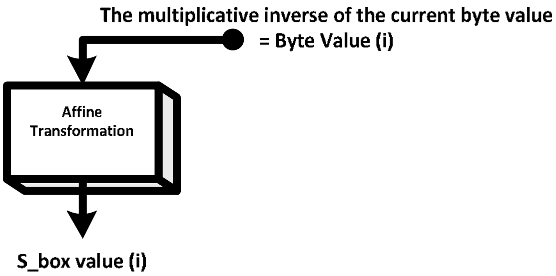
In matrix form, the affine transformation element of the S-box can be expressed as equation (10):

$$\begin{bmatrix} b_0' \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{(10)}$$

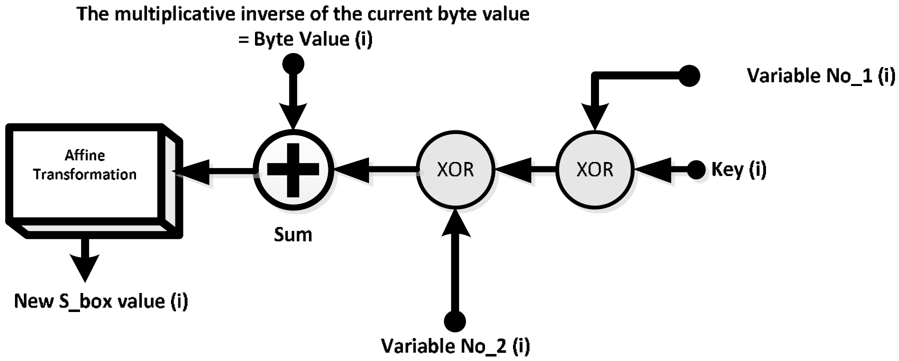
3. Proposed Algorithm

A single 128-bit block will be the input to the encryption and decryption algorithms where this block is depicted as a square matrix of bytes, and will then be copied into the state array, which will be modified at each stage of encryption or decryption. The encryption and decryption process of this new design resembles the original AES, it has confusion and diffusion layer and key addition layer as well.

Figure (5a) shows the original algorithm of S-box. The proposed algorithm is described in **Figure (5b).** An S-box is a one to one mapping for all byte values from 0 to 255. The S-box is used to change the original plain text in bytes to cipher text. All values are represented in hexadecimal notation.



(a)



(b)

Fig .(5) (a) Original algorithm of S_box (b) Proposed algorithm of new S_box.

The proposed AES is used algorithm to create a new S-box and Inverse S-box as shown in **Figures (6)** and **(7).** Each value in old S-box matrix is modified with a new value to create a new S-box matrix. In this paper, three keys are proposed (Key (i), Variable No._1 (i) and Variable No._2 (i)) for ciphering.

6a 75 6c d9 e9 81 3c 2f 63 1a bb 22 d3 de a2 7f
68 af 15 66 e1 85 4e eb b6 08 f1 b4 cf ad 21 ed
6b ae 88 84 dc 24 2b d7 3d be fe 1b 5c d1 38 1c
d8 ce c9 90 ba 4a 0c 93 a5 3f 89 b1 49 fb a9 6e
12 d0 01 13 c7 43 f8 f3 5b 32 0a 59 c3 ea 7c 9f
00 ca 09 f6 3b f5 b8 87 71 17 b7 9b a0 1f 51 25
3a f4 76 11 58 44 1e 8c e7 f0 e8 64 7d d6 96 74
02 8e aa 86 78 b0 e4 29 91 14 30 28 19 ac fa db
c4 5f 40 e5 b5 35 98 fd df bc 94 26 c6 54 c5 7a
7b d2 62 8f 39 31 99 2a 9a 04 a3 0f 8d 45 10 c0
b3 61 e6 03 52 da 2d 47 cb c8 46 79 8a 9c 0e 70
fc c1 2c 3e 67 77 1d 0b 65 4d 56 36 b9 a6 a7 e2
a1 a4 f9 27 07 4c bd dd 34 d4 9e 16 50 ee 57 83
23 37 69 6f 41 18 ff 5d 8b 2e 5e 53 9d 92 f7 cd
cc e3 72 42 60 c2 95 48 80 05 6d e0 d5 bf 33 f2
97 a8 55 20 ec ef 4b 73 5a 82 7e 06 ab 4f b2 0d

Fig .(6) New S-Box.

50 42 70 a3 99 e9 fb c4 19 52 4a b7 36 ff ae 9b
9e 63 40 43 79 12 cb 59 d5 7c 09 2b 2f b6 66 5d
f3 1e 0b d0 25 5f 8b c3 7b 77 97 26 b2 a6 d9 07
7a 95 49 ee c8 85 bb d1 2e 94 60 54 06 28 b3 39
82 d4 e3 45 65 9d aa a7 e7 3c 35 f6 c5 b9 16 fd
cc 5e a4 db 8d f2 ba ce 64 4b f8 48 2c d7 da 81
e4 a1 92 08 6b b8 13 b4 10 d2 00 20 02 ea 3f d3
af 58 e2 f7 6f 01 62 b5 74 ab 8f 90 4e 6c fa 0f
e8 05 f9 cf 23 15 73 57 22 3a ac d8 67 9c 71 93
33 78 dd 37 8a e6 6e f0 86 96 98 5b ad dc ca 4f
5c c0 0e 9a c1 38 bd be f1 3e 72 fc 7d 1d 21 11
75 3b fe a0 1b 84 18 5a 56 bc 34 0a 89 c6 29 ed
9f b1 e5 4c 80 8e 8c 44 a9 32 51 a8 e0 df 31 1c
41 2d 91 0c c9 ec 6d 27 30 03 a5 7f 24 c7 0d 88
eb 14 bf e1 76 83 a2 68 6a 04 4d 17 f4 1f cd f5
69 1a ef 47 61 55 53 de 46 c2 7e 3d b0 87 2a d6

Fig .(7) New Inverse S-Box.

The two different numbers (Variable No._1, Variable No._2) are a random numbers and it is changed at each value of S-box. The constant byte value (Key) is also secret number and it is a XORed with the first variable number (Variable No._1). The result value is then XORed with the second variable number (Variable No._2).

The final value will combine with the multiplicative inverse of the current byte value with respect to the specified modulo polynomial transformed (by using affline transformation). The result value is the new S-box value and it's used by SubBytes substitution.

In order to recover the data previously encrypted it is required to perform the inverse of the new S-box and then complete other decryption process to get the original data. It is very difficult to decrypt the cipher text without using the correct key of S-box.

The flow chart of the proposed algorithm is shown in **Figure (8).**

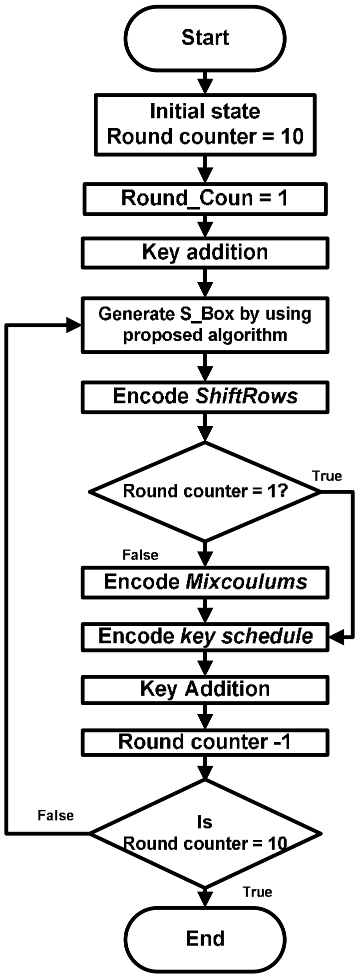


Fig .(8) Flow chart of the proposed AES algorithm.

4. Simulation Results

The plaintext is a 16 byte and in hexadecimal format (plaintext_Hex_No. = {**11 45 88 cf ff a4 99 dd ee 6e 1a 3e 11 7f b3 f7**}). The cipher key length and data block size in AES is 16byte/128 bits. The simulation results shows the data, key used, cipher (encrypted data) and decrypted data, as shown in **Tables (2) And (3)** respectively, the values of the original and calculated plaintext values are in agreement, thereby proving the correctness of the design's operation.

It can be noted that the plaintext in **Tables (2) And (3)** are the same on a per bit basis, thereby proving the working of the proposed algorithm. Also the ciphertext obtained from program, as shown in **Table (2),** is compared with the ciphertext shown in **Table (3).** The aim of this exercise is to prove that the encryption and decryption are in fact in keeping with the Rijndael specifications, and that the results obtained are not merely obtained because encryption and decryption are the inverse of each other. The randomly key-dependent S-boxes make our approach resistant to linear and differential cryptanalysis. This approach will lead to generate more secure block ciphers, solve the problem of the fixed structure S-boxes, and will increase the security level of the AES block cipher system. The main advantage of such approach is that an enormous number of S-boxes can be generated by changing secret key.

Table (2) Encrypting of 16 Byte Plaintext.

	Proposed AES	Original AES
Plaintext (16byte):	11 45 88 cf ff a4 99 dd ee 6e 1a 3e 11 7f b3 f7	11 45 88 cf ff a4 99 dd ee 6e 1a 3e 11 7f b3 f7
Key:	00 01 02 03 04 05 06 0708 09 0a 0b 0c 0d 0e 0f	00 01 02 03 04 05 06 0708 09 0a 0b 0c 0d 0e 0f
Round 1:	00 04 08 0c 01 05 09 0d 02 06 0a 0e 03 07 0b 0f	00 04 08 0c 01 05 09 0d 02 06 0a 0e 03 07 0b 0f
Round 2:	df db d3 df a3 a6 af a2 7d 7b 71 7f d0 d7 dc d3	d6 d2 da d6 aa af a6 ab 74 72 78 76 fd fa f1 fe
Round 3:	3b e0 33 ec 78 de 71 d3 12 69 18 67 1d ca 16 c5	b6 64 be 68 92 3d 9b 30 cf bd c5 b3 0b f1 00 fe
Round 4:	50 b0 83 6f f4 2a 5b 88 5e 37 2f 48 c8 02 14 d1	b6 d2 6c 04 ff c2 59 69 74 c9 0c bf 4e bf bf 41
Round 5:	87 37 b4 db af 85 de 56 69 5e 71 39 bc be aa 7b	47 95 f9 fd f7 35 6c 05 f7 3e 32 8d bc 03 bc fd
Round 6:	2f 18 ac 77 90 15 cb 9d 41 1f 6e 57 ef 51 fb 80	3c a9 50 ad aa 9f f3 f6 a3 9d af 22 e8 eb 57 aa
Round 7:	4a 52 fe 89 17 02 c9 54 85 9a f4 a3 c6 97 6c ec	5e f7 a7 0a 39 a6 55 a3 0f 92 3d 1f 7d 96 c1 6b
Round 8:	31 63 9d 14 14 16 df 8b 50 ca 3e 9d 7a ed 81 6d	14 e3 44 4e f9 5f 0a a9 70 e2 df c0 1a 8c 4d 26
Round 9:	97 f4 69 7d 51 47 98 13 86 4c 72 ef 9b 76 f7 9a	47 a4 e0 ae 43 1c 16 bf 87 65 ba 7a 35 b9 f4 d2
Round 10:	ea 1e 77 0a a3 e4 7c 6f 25 69 1b f4 37 41 b6 2c	54 f0 10 be 99 85 93 2c 32 57 ed 97 d1 68 9c 4e
Ciphertext (16byte):	65 8a 6a c0 c6 49 57 b4 6f 10 5f f7 f5 22 f5 77	b2 62 01 d4 2b 08 a7 fd b1 1a 00 fa da 82 19 a4

Table (3) Decrypting of 16 Byte Ciphertext.

	Proposed AES	Original AES
Ciphertext (16byte):	65 8a 6a c0 c6 49 57 b4 6f 10 5f f7 f5 22 f5 77	b2 62 01 d4 2b 08 a7 fd b1 1a 00 fa da 82 19 a4
Key:	00 01 0203 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	00 01 0203 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Round 1:	a8 b6 c1 cb 4f ab d7 b8 79 10 0b ff 8c cd 7b 57	54 f0 10 be 99 85 93 2c 32 57 ed 97 d1 68 9c 4e
Round 2:	ea 1e 77 0a a3 e4 7c 6f 25 69 1b f4 37 41 b6 2c	47 a4 e0 ae 43 1c 16 bf 87 65 ba 7a 35 b9 f4 d2
Round 3:	97 f4 69 7d 51 47 98 13 86 4c 72 ef 9b 76 f7 9a	14 e3 44 4e f9 5f 0a a9 70 e2 df c0 1a 8c 4d 26
Round 4:	31 63 9d 14 14 16 df 8b 50 ca 3e 9d 7a ed 81 6d	5e f7 a7 0a 39 a6 55 a3 0f 92 3d 1f 7d 96 c1 6b
Round 5:	4a 52 fe 89 17 02 c9 54 85 9a f4 a3 c6 97 6c ec	3c a9 50 ad aa 9f f3 f6 a3 9d af 22 e8 eb 57 aa
Round 6:	2f 18 ac 77 90 15 cb 9d 41 1f 6e 57 ef 51 fb 80	47 95 f9 fd f7 35 6c 05 f7 3e 32 8d bc 03 bc fd
Round 7:	87 37 b4 db af 85 de 56 69 5e 71 39 bc be aa 7b	b6 d2 6c 04 ff c2 59 69 74 c9 0c bf 4e bf bf 41
Round 8:	50 b0 83 6f f4 2a 5b 88 5e 37 2f 48 c8 02 14 d1	b6 64 be 68 92 3d 9b 30 cf bd c5 b3 0b f1 00 fe
Round 9:	3b e0 33 ec 78 de 71 d3 12 69 18 67 1d ca 16 c5	d6 d2 da d6 aa af a6 ab 74 72 78 76 fd fa f1 fe
Round 10:	df db d3 df a3 a6 af a2 7d 7b 71 7f d0 d7 dc d3	00 04 08 0c 01 05 09 0d 02 06 0a 0e 03 07 0b 0f
Plaintext (16byte):	11 45 88 cf ff a4 99 dd ee 6e 1a 3e 11 7f b3 f7	11 45 88 cf ff a4 99 dd ee 6e 1a 3e 11 7f b3 f7

4.1 Avalanche Test

The avalanche effect property is very important for encryption algorithm. This property can be seen when changing one bit in plaintext and then watching the change in the outcome of at least half of the bits in the ciphertext. One purpose for the avalanche effect is that by changing only one bit there is large change then it is harder to perform an analysis of ciphertext, when trying to come up with an attack. Avalanche Effect can be calculated by using equation (11) [16].

$$Avalanche\_Effect = \frac{Number\_of\_flipped\_bits\_in\_ciphertext}{Number\_of\_bits\_in\_ciphertext} \quad \ldots\ldots\ldots\ldots\ldots\ldots(11)$$

A block cipher is said to have a poor randomization if it does not exhibit the avalanche effect to a significant degree. For a good quality block cipher, such a small change in either key or plaintext should cause a drastic change in the ciphertext.

To perform the test we change plaintext bit to “01” instead of “11” and “80” instead of “00” the result obtained is 0.5468 and 0.5234 respectively, which prove that proposed AES pass avalanche test. The results of avalanche effect are given in **Tables (4) And (5)** respectively. The proposed AES has good avalanche test if compared with results of original AES and proposed AES in Ref.[9]. This avalanche result reflects the immunity of our algorithm to linear and differential cryptanalysis. Avalanche test is important features for strong S-boxes to produce more confusion to the encryption process.

Table (4) The results of avalanche test due to one bit change in plaintext (From (11) to (01) in plaintext).

	Algorithm	Input Data (plaintext)	Output Data(ciphertext)	Avalanche Test
Cipher Key = '0123456789ABCDEF'	This Work	11,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	D4,68,1B,28,83,92,E7,C0, F0,B0,BD,CE,B0,18,ED,07	0.5468
		01,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	0A,A1,B8,13,60,F5,5A,A9, 31,FF,B3,1A,E4,8F,62,80	
	AES-RC4 [9]	11,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	98,AC,21,A7,EF,17,17,16, BF,CB,B6,8E,B8,5E,7F,C8	0.5078
		01,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	96,63,13,AE,F5,B4,48,52, B6,30,18,7A,BB,C8,62,C2	
	Original AES [9]	11,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	1D,BA,92,8F,49,FC,58,29, B8,ED,48,96,4E,74,01,E4	0.5078
		01,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	FA,B6,58,C6,07,81,EE,E4, 5D,43,40,B5,1D,7E,BE,FC	

Table (5) The results of avalanche test due to one bit change in plaintext (From (00) to (80) in plaintext).

	Algorithm	Input Data (plaintext)	Output Data(ciphertext)	Avalanche Test
Cipher Key = '0123456789ABCDEF'	This Work	00,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	4D,6D,A8,97,1C,4F,A9,4F, 1b,CE,2F,04,92,E4,4F,90	0.5234
		80,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	93,A9,89,BC,E5,BA,5a,2E, 05,26,F8,98,31,D6,6D,E4	
	AES-RC4 [9]	00,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	A9,23,41,A8,1F,FF,EB,80, 07,F3,C0,61,79,53,CA,EE	0.5234
		80,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	3C,C3,19,3C,F0,B6,3D,E9, 3B,96,EC,9E,3B,C0,FD,7C	
	Original AES [9]	00,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	7D,F7,6B,0C,1A,B8,99,B3, 3E,42,F0,47,69,1B,54,6F	0.4688
		80,00,00,00,00,00,00,00, 00,00,00,00,00,00,00,00	F6,C7,1E,ED,C3D9,96,B1, 83,CB,56,8D,15,68,EB,06	

5. Conclusions

This paper presents an optimized S-Box for Advanced Encryption Standard (AES) design. The S-Box is one of the most important components of AES. During Subbyte transformation, the eight bit input is substituted by eight bit output using the S-Box. S-Box is constructed by composing two transformation- multiplicative inverse in Galois Field GF(2^8) followed by an affine transformation. The proposed algorithm can be implemented without increasing the size of the key block. Though the original algorithm is very secure, the proposed changes in the processing of the algorithm will help to encrypt the data by making stronger diffusion and confusion. It also increases the complexity of the algorithm multiple times. The achieved result gives more resistance to Brute-force attack and will make it very difficult to decrypt the plaintext without correct values of S-box matrix. AES is a strong algorithm due to its large rounds and algebraic complexity inside the rounds. The proposed method has been shown improved security of proposed AES over the traditional AES by increasing the complexity of encryption process. The main advantage of this algorithm is that many S-Boxes can be generated by changing cipher key. The proposed algorithm leads to increase the complexity of encryption process and makes the differential and linear cryptanalysis more difficult.

6. References

1. **P.Karthigaikumar and Soumiya Rasheed, "Simulation of Image Encryption using AES Algorithm", IJCA, Special Issue on Computational Science - New Dimensions & Perspectives, pp. 166-172, 2011.**
2. **Ingrid Verbauwhede, Patrick Schaumont and Henry Kuo, "Design and Performance Testing of a 2.29-GB/s Rijndael Processor'', IEEE Journal of Solid-State Circuits, Vol.38, No.3, pp.569-572, 2003.**
3. **Kazys KazlAaussKkas, Jaunius Kazlauskas, "Key-Dependent S-Box Generation in AES Block Cipher System'', Informatics, Institute of Mathematics and Informatics, Vol. 20, No. 1, pp. 23–34, 2009.**
4. **Orlando J. Hernandez, etc. al, "A Low Cost Advanced Encryption Standard (AES) Co-Processor Implementation", Journal of Computer Science & Technology (JCS&T), Vol.8, No.1, pp. 8-14, 2008.**
5. **Rashi Kohli etc. al, "S-Box Design Analysis and Parameter Variation in AES Algorithm", International Journal of Computer Applications, Vol.60, No.2, December 2012.**
6. **K.Rahimunnisa, etc. al, " Implementation of AES with New S-Box and Performance Analysis with the Modified S-Box", International Journal of Computer Applications (IJCA), International Conference on VLSI, Communication & Instrumentation (ICVCI), Vol.2, pp.5-8, 2011.**
7. **P.Saravanan etc. al, " A High-Throughput ASIC implementation of Configurable Advanced Encryption Standard (AES) Processor", International Journal of Computer Applications (IJCA), Vol. Network Security and Cryptography (NSC), Vol.3, pp.1-6, 2011.**
8. **L.Thulasimani, etc. al, "Design and Implementation of Reconfigurable Rijndael Encryption Algorithms for Reconfigurable Mobile Terminals", International Journal on Computer Science and Engineering (IJCSE), Vol.2, No.4, pp. 1003-1011, 2010.**
9. **I. Abd ElGhafar, etc. al, "Generation of AES Key Dependent S-Boxes using RC4 Algorithm", 13th International Conference on Aerospace Sciences & Aviation Technology (ASAT), CE.24, pp.1-9, 2009.**
10. **Razi Hosseinkhani and H. Haj Seyyed Javadi, "Using Cipher Key to Generate Dynamic S-Box in AES Cipher System", International Journal of Computer Science and Security (IJCSS), Vol.6, No.1, pp.19-28, 2012.**

11. **Jie Cui, etc. al, "An Improved AES S-box And Its Performance Analysis", International Journal of Innovative Computing, Information and Control, Vol. 7, No.5, pp. 2291-2302, May 2011.**
12. **Chandrasekharappa T.G.S., etc. al, "S-boxes generated using Affine Transformation giving Maximum Avalanche Effect", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 9, pp.3185-3193, September 2011.**
13. **J. Daemen and V. Rijmen, "AES Proposal: Rijndael", The Computer Security Resource Center (CSRC), Division of National Institute of Standards and Technology (NIST),September 1999, Available online: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf.**
14. **E. M. Mahmoud, etc. all, "Dynamic AES-128 with Key-Dependent S-box", International Journal of Engineering Research and Applications (IJERA), Vol. 3, No.1, pp.1662-1670, January -February 2013.**
15. **Julia Juremi, etc. all, "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.**
16. **Akash K. Mandal and Archana Tiwari, "Analysis of Avalanche Effect in Plaintext of DES using Binary Codes", International Journal of Emerging Trends and Technology in Computer Science (IJETTCS), Vol.1, No.3, pp. 166-177, 2012.**