Substitution Box(S-Box) Implementation Based On Advanced Encryption Standard

A Research Project Report Submitted In Partial FulfillmentOf The Requirements For The Degree Of

M.sc in Computer Science

By

Md. Aktaruzzaman
Student Id: CSE202003055
24th Batch
Dept. of CSE, JU.
Session: 2020-2021

Supervised by

Dr. Abu Sayed Md.Mosafizur Rahaman
Professor
Department of Computer Science & Engineering
Jahangirnagar University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAHANGIRNAGAR UNIVERSITY

February, 2022

JAHANGIRNAGAR UNIVERSITY

SAVAR, BANGALADESH

CERTIFICATE

Th***is is to certify that the Research Project*** entitled,"Substitution Box(S-Box) Implementation Based On Advanced Encryption Standard''submitted by "Md.Aktaruzzaman" in partial fulfillment of the requirements for the award of M.sc Degree in Computer Science at the Jahangirnagar University,Bangladesh is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the Research Project has not been submitted to any other university / institute for the award of any Degree or Diploma.

DATE:

Dr. Abu Sayed Md.Mosafizur Rahaman
Professor
Department of Computer Science & Engineering
Jahangirnagar University

JAHANGIRNAGAR UNIVERSITY

SAVAR, BANGALADESH

ACKNOWLEDGEMENT

I am thankful to Dr. Abu Sayed Md.Mosafizur Rahaman**,** Professor in the department of Computer Science and Engineering,JU,Bangladesh for giving me the opportunity to work under him and lending every support at every stage of this project work.

I would also like to convey my sincerest gratitude and ineptness' to all other faculty members and staff of Department of Computer Science and Engineering,JU,Bangladesh, who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish the project work.

Date:

Md.Aktaruzzaman
Dept. of Computer & Science
Jahangirnagar University,Bangladesh
ID-CSE202003055

CONTENTS

ABSTRACT:

On October, 2, 2000, The National Institute of Standards and Technology (NIST) announced Rijndael as the new Advanced Encryption Standard (AES).The Predecessor to the AES was Data Encryption Standard (DES) which was considered to be insecure because of its vulnerability to brute force attacks. DES was a standard from 1977 and stayed until the mid 1990's. However, by the mid 1990s, it was clear that the DES's 56-bit key was no longer big enough to prevent attacks mounted on contemporary computers, which were thousands of times more powerful than those available when the DES was standardized. The AES is a 128 bit Symmetric block Cipher.

This Research Project includes the complete step by step implementation of Advanced Encryption Technique, i.e. encrypting and decrypting 128 bit data using the AES and it's modification for enhanced reliability and security. The encryption process consists of the combination of various classical techniques such as substitution, rearrangement and transformation encoding techniques. The encryption and decryption modules include the Key Expansion module which generates Key for all iterations. The modifications include the addition of an arithmetic operation and a route transposition cipher in the **attacks** iterative rounds. The Key expansion module is extended to double the number of iterative processing rounds in order to increase its immunity against unauthorized attacks**.**

Introduction to cryptography:

Cryptography is the science of information and communication security. Cryptography is the science of secret codes, enabling the confidentiality of communication through an insecure channel. It protects against unauthorized parties by preventing unauthorized alteration of use. It uses an cryptographic system to transform a plaintext into a cipher text, using most of the time a key.

There exists certain cipher that doesn't need a key at all. An example is a simple Caesar-cipher that obscures text by replacing each letter with the letter thirteen places down in the alphabet. Since our alphabet has 26 characters, it is enough to encrypt the cipher text again to retrieve the original message.

Introduction to the Advanced Encryption Standard:

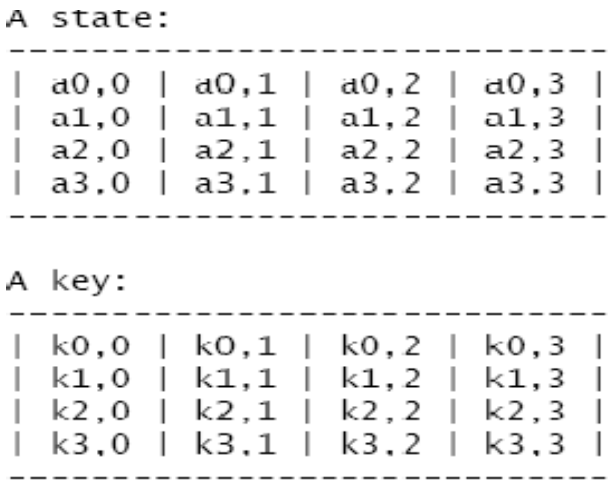
The Advanced Encryption Standard, in the following referenced as AES, is the winner of the contest, held in 1997 by the US Government, after the **Data Encryption Standard** was found too weak because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted in 1998 and based on public comments the pool was reduced to five finalists in 1999. In October 2000, one of these five algorithms was selected as the forthcoming standard: a slightly modified version of the Rijndael.

The Rijndael, whose name is based on the names of its two Belgian inventors, **Joan Daemen** and **Vincent Rijmen**, is a **Block cipher**, which means that it works on fixed-length group of bits, which are called *blocks*. It takes an input block of a certain size, usually 128, and produces a corresponding output block of the same size. The transformation requires a second input, which is the secret key. It is important to know that the secret key can be of any size (depending on the cipher used) and that AES uses three different key sizes: 128, 192 and 256 bits.

While AES supports only block sizes of 128 bits and key sizes of 128, 192 and 256 bits, the original Rijndael supports key and block sizes in any multiple of 32, witha minimum of 128 and a maximum of 256 bits.

Description of the Advanced Encryption Standard algorithm

AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called *state*. The state is a rectangular array of bytes and since the block size is 128 bits,which is 16 bytes, the rectangular array is of dimensions 4x4. (In the Rijndael version with variable block size, the row size is fixed to four and the number of columns varies. The number of columns is the block size divided by 32 and denoted **Nb**). The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted **Nk**, is equal to the key length divided by 32.



It is very *important* to know that the cipher input bytes are mapped onto the state bytes in the order a0,0, a1,0, a2,0, a3,0, a0,1, a1,1, a2,1, a3,1 ... and the bytes of thecipher key are mapped onto the array in the order k0,0, k1,0, k2,0, k3,0, k0,1, k1,1,k2,1, k3,1 ... At the end of the cipher operation, the cipher output is extracted from the state by taking the state bytes in the same order. AES uses a variable number ofrounds, which are fixed: A key of size 128 has 10 rounds. A key of size 192 has 12rounds. A key of size 256 has 14 rounds.

<u>During each round, the following operations are applied on the state</u>:

Sub Bytes: every byte in the state is replaced by another one, using the Rijndael S-Box
Shift Row: every row in the 4x4 array is shifted a certain amount to the left
Mix Column: a linear transformation on the columns of the state
Add Round Key: each byte of the state is combined with a round key, which is a Different key for each round and derived from the Rijndael key schedule.

<u>Salient Features:</u>

- The cipher key is expanded into a larger key, which is later used for theactual operations
- The round Key is added to the state before starting the with loop
- The **Final Round ()** is the same as **Round (),** apart from missing the **MixColumns ()** operation.
- During each round, another part of the **Expanded Key** is used for theoperations
- The Expanded Key shall always be derived from the Cipher Key and neverbe specified directly.

<u>AES operations: SubBytes, ShiftRow, MixColumn and AddRoundKey</u>

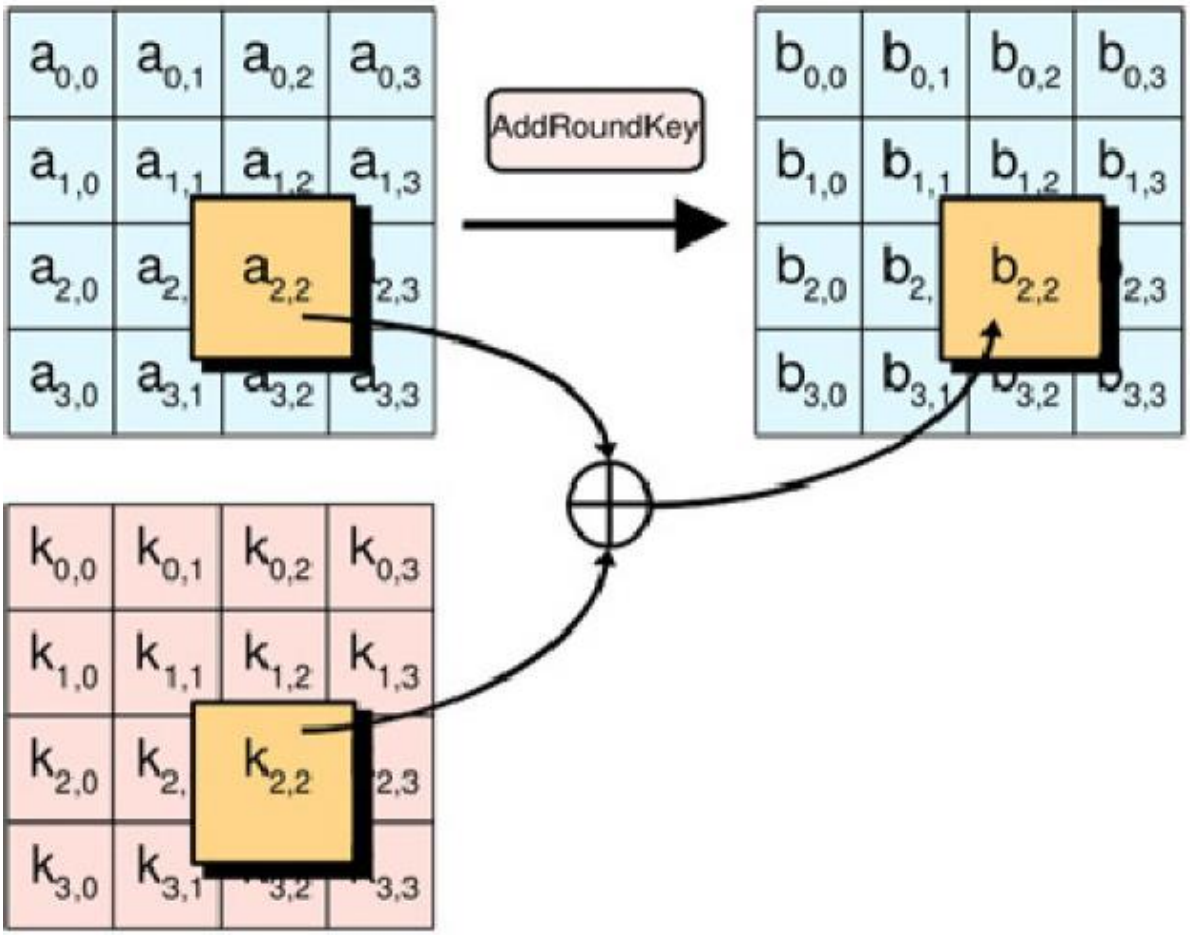
- **The Add Round Key operation:**

In this operation, a Round Key is applied to the state by a simple bitwise XOR.The Round Key is derived from the Cipher Key by the means of the key schedule. The Round Key length is equal to the block key length (=16 bytes).

a0,0	a0,1	a0,2	a0,3		k0,0	k0,1	k0,2	k0,3		b0,0	b0,1	b0,2	b0,3
a1,0	a1,1	a1,2	a1,3	XOR	k2,0	k2,1	k2,2	k2,3	=	b2,0	b2,1	b2,2	b2,3
a2,0	a2,1	a2,2	a2,3		k1,0	k1,1	k1,2	k1,3		b1,0	b1,1	b1,2	b1,3
a3,0	a3,1	a3,2	a3,3		k3,0	k3,1	k3,2	k3,3		b3,0	b3,1	b3,2	b3,3

where: b(i,j) = a(i,j) XOR k(i,j)

A graphical representation of this operation can be seen below:



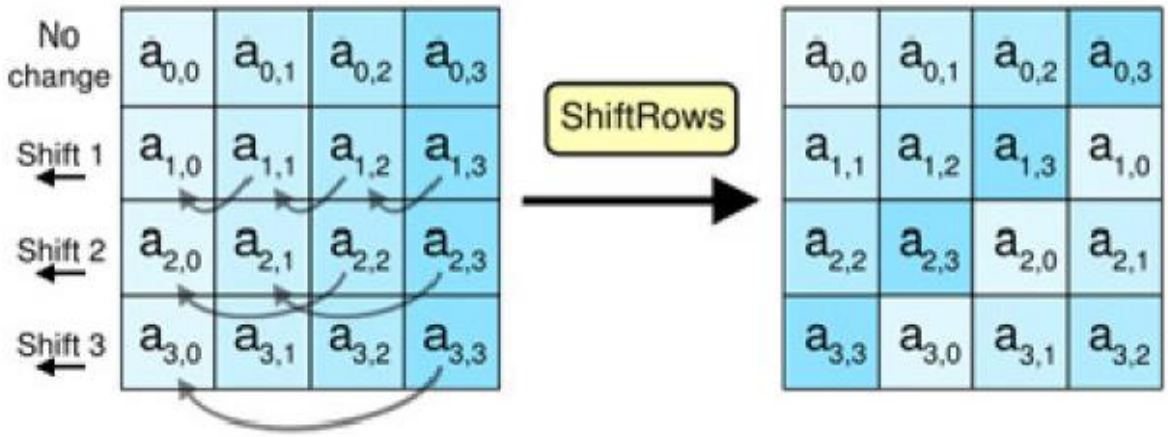
- ***<u>The Shift Row operation:</u>***

In this operation, each row of the state is cyclically shifted to the left, dependingon the row index.

The 1st row is shifted 0 positions to the left. The 2nd row is shifted 1 position to the left. The 3rd row is shifted 2 positions to the left. The 4th row is shifted 3 positions to the left.

```
-----------------------------     -----------------------------
| a0,0 | a0,1 | a0,2 | a0,3 |     | a0,0 | a0,1 | a0,2 | a0,3 |
| a1,0 | a1,1 | a1,2 | a1,3 | ->  | a1,1 | a0,2 | a1,3 | a1,0 |
| a2,0 | a2,1 | a2,2 | a2,3 |     | a2,2 | a2,3 | a2,0 | a2,1 |
| a3,0 | a3,1 | a3,2 | a3,3 |     | a3,3 | a3,0 | a3,1 | a3,2 |
-----------------------------     -----------------------------
```

A graphical representation of this operation can be found below:



The inverse of **Shift Row** is the same cyclically shift but to the right. Itwill be needed later for decoding.

- ***The Sub Bytes operation*:**

The **Sub Bytes** operation is a non-linear byte substitution, operating on each byte of the state independently. The **substitution table (S-Box)** is invertible and is constructed by the composition of two transformations:

1. Take the multiplicative inverse in **Rijndael's finite field**
2. Apply an affine transformation which is documented in the Rijndaeldocumentation.

Since the S-Box is independent of any input, pre-calculated forms are used. Each byte of the state is then substituted by the value in the S-Boxwhose index corresponds to the value in the state:

$$a(i,j) = SBox[a(i,j)]$$

The inverse of SubBytes is the same operation, using the inversed S-Box, which isalso precalculated.

- ***The MixColumn operation:***

This section involves advance mathematical calculations in the **Rijndael's finitefield**. It corresponds to the matrix multiplication with:

```
2  3  1  1

1  2  3  1

1  1  2  3

3  1  1  2
```

And that the addition and multiplication operations are different from the normalones.

- ***The Rijndael Key Schedule***

The Key Schedule is responsible for expanding a short key into a larger key, whose parts are used during the different iterations. Each key size is expanded to a different size:

An 128 bit key is expanded to an 176 byte key.
An 192 bit key is expanded to an 208 byte key.
An 256 bit key is expanded to an 240 byte key.

There is a relation between the cipher key size, the number of rounds and the Expanded Key size. For an 128-bit key, there is one initial Add Round Key operation plus there are 10 rounds and each round needs a new 16 byte key, therefore we require 10+1 Round Keys of 16 byte, which equals 176 byte. Thesame logic can be applied to the two other cipher key sizes. The general formula isthat:

```
Expanded Key Size = (nbrRounds+1) * BlockSize
```

- *Rotate:*

The 4-byte word is cyclically shifted 1 byte to the left:

```
--------------------    --------------------
| 1d | 2c | 3a | 4f | -> | 2c | 3a | 4f | 1d |
--------------------    --------------------
```

- ***S-Box:***

The S-Box and Inverse S-Box in the tradition advance encryption standard is fixed and it is made by the composite field arithmetic to find the multiplicative inverse in the finite field GF (28). But in traditional AES the use of irreducible polynomial m(x) = x8+ x4 + x3 + x+1 to find out multiplicative inverse, is known to the attacker. So if we can make use of different irreducible polynomial every time to the finite field of GF (28) and send this the receiver combined with the secret key, then every time a new irreducible polynomial is used and a random S-Box is generated. Hence, the security of the algorithm is enhanced. In this paper, we devise an algorithm, which make use of different irreducible polynomial to finite field of GF (28) to make random S-Box and Inverse S-Box .Keywords-Substitute bytes, S-Box, Inverse S- Box Multiplicative Inverse, Irreducible Polynomial, Finite field.

- ***Implementation: S-Box***

The S-box in AES algorithm, which is invertible, is constructed by composing two transformations:

1) Initialize the S-box with the byte values in ascending sequence row by row. The first row contains {00}, {01},{02}... {0F}; the second row contains {10}, {11}, etc.; and so on. Thus, the value of the byte at row x, column y is{x y}. 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.

2) Consider that each byte in the S-box consists of 8 bitslabeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the following transformation to each bit of each byte in the S-box.

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

Where c_i is the i th bit of byte c with the value {63}; that is, ($c_7c_6c_5c_4c_3c_2c_1c_0$) = (01100011). The prime (') shown belowindicates that the variable is to be updated by the value on the right. The AES standard depicts this transformation in matrix form as follows:

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \text{---(2)}$$

Equation (1) has to be interpreted carefully. In ordinary matrix multiplication, each element in the product matrix is the sum of products of the elements of one row and one column. In this case, each element in the product matrix is the bitwise XOR of products of the elements of one row and one column. Further, the final addition shown in Equation (2) is a bitwise XOR.

The inverse substitute byte transformation makes use of the inverse S-box. Note, for example, that the input {2A} produces the output {95} and the input {95} to the S-box produces {2A}. The inverse S-box is constructed by applying the inverse of the transformation in Equation (3) followed by taking the multiplicative inverse in GF (2^8).

The inverse transformation is:

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i \quad \text{.....(3)}$$

Algorithm Of AES S-Box And Inverse AES S-Box

```
let irreducible_poly = "1 + x + x**3 + x**4 + x**8";
let p;
p = parseInt(eval(irreducible_poly.replace(/x/g, '10')), 2) // Parse it to Decimal value

// Calculate Multiplicative Inverse
1. let t = new Uint32Array(256);
2. let i=0,x=1;
3. for i=0 to 256
4.  t[i] = x;
5. x ^= (x << 1) ^ ((x >>> 7) * p)


// Generate Sbox with Affine Transformation
1. let Sbox = new Uint32Array(256);
2. Sbox[0] = 0x63;
3. for i=0 to 256
4. let x = t[255 - i];
5. x |= x << 8;
6. x ^= (x >> 4) ^ (x >> 5) ^ (x >> 6) ^ (x >> 7);
7. Sbox[t[i]] = (x ^ 0x63) & 0xFF;
8. return Sbox;


// Inverse of Sbox
1. function inverse(sbox)
2. let InvSbox = new Uint32Array(256);
3. for i=0 to 256
5. InvSbox[i] = sbox.indexOf(i);
6. return InvSbox;
```

Input :

S-Box Generator

Polynomial : 1 + x + x**3 + x**4 + x**8

Generate

S-Box Inverse S-Box

Output:

S-Box Generator

Polynomial : 1 + x + x**3 + x**4 + x**8

Generate

S-Box

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

Inverse S-Box

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

Complexity Of AES S-Box

Time O(1) Memory O(1) We can implement S box with a lookup, or as a circuit or recreate the original DF polynomial used to generate it. In all cases it's a fixed size, it wouldn't be AES otherwise. And with no n. to discuss the time and memory are constant.

As a more general note complexity notations are problematic in cryptography, also breaking AES via brute force is technically O(1). So we often talk about number of operations, and say things like 2^256 operations, while often keeping it vague what the operation is. Sometimes it is invocation of the cryptographic primitive which makes it clear and we can count, but often we do something else and what is the operation’s we are counting becomes murky.

CONCLUSION:

The Advanced Encryption Technique(S-Box) was implemented successfully using 'Html,CSS,JavaScript' language. The proposed modified S-Box and Inverse S- Box is very useful to protect the AES algorithm from the attacker. Though, AES is not yet broken by any attacker, but the sole concept of the random S-Box generation and making use of different irreducible polynomials every time, makes the algorithm stronger. The only thing is to send the irreducible polynomial with the key to the receiver, so that the receiver can also calculate the S-Box secretly. So it enhances the security of AES algorithm.

REFERENCES:

[1] https://crypto.stackexchange.com/

[2] https://en.wikipedia.org/wiki/Rijndael_S-box.

[3] https://www.redalyc.org/journal/5122/512253718012/html/

[4] https://pdf.sciencedirectassets.com/

[5] https://cryptography.fandom.com/wiki/Rijndael_S-box

[6] https://www.researchgate.net/publication/261429022_Random_S-Box_generation_in_AES_by_changing_irreducible_polynomial

[7] https://www.scirp.org/journal/paperinformation.aspx?paperid=92179