Double S-Box for Data Encryption and Decryption with Modified Advanced Encryption Standard

Sunil Kishor Khode[1], Dr. Manish. P. Deshmukh[2]
[1]Research Scholar,
SSBT's COET, Bambhori, Jalgaon, India, Pin-425001
[2]Professor & Head E & TC Engg. Dept,
SSBT's COET, Bambhori, Jalgaon, India, Pin-425001

Abstract: One of the most popular and extensively used symmetric block cypher algorithms in the world is the Advanced Encryption Standard (AES) method.This method, which is used in hardware and software worldwide, has a unique structure for encrypting and decrypting sensitive data. When using the AES algorithm to encrypt data, it is very difficult for hackers to obtain the actual data. There is currently no proof that this algorithm is broken. Each of these cyphers has a block size of 128 bits, and AES can handle three alternative key sizes: AES 128-, 192-, and 256-bit. An overview of the AES algorithm is given in this work, along with a detailed discussion of several of its key properties and examples of earlier research on the method. Our approach offers a high level of security by utilising a modified AES with Double S-Box. The results show that DS-Box MAES is 0.90 times quicker than AES and passed the Monte Carlo test. Additionally, DS-Box MAES offers a greater avalanche effect than AES, making it more resilient to modifications in keys or plaintext.

Keywords: S-Box, DS-Box, encryption, decryption, cryptography, and AES (Advanced Encryption Standard).

I.Overview

Cryptographic techniques have been widely employed in recent years to combat security risks (Stallings, 2010) [1]. These days, secure systems are quite important. All kinds of solutions, from highly parallel high performance computing processors to area-sensitive embedded sensors, require secure data transfer and storage. We investigated various implementation strategies for the Advanced Encryption Standard (AES) encryption algorithm because of the wide range of area and performance requirements.

Many cryptographic applications today employ AES, an example of a symmetric key cryptographic algorithm. In 2001, the ageing Data Encryption Standard (DES) algorithm was replaced by the new commercial cryptographic method, AES (sometimes called the Rijndael cypher algorithm) (Advanced Encryption Standard, 2001) [2].

Because it is faster, more secure, and uses less power than its software-based counterpart, the hardware-based implementation of the AES algorithm is crucial. Field Programmable Gate Arrays (FPGAs) are a desirable option for hardware-based AES implementation (Saggese et al., 2003; Elbirt et al., 2001). [3] through [4]. Originally, FPGAs were employed as glue logic. But in recent years, their market and capabilities have grown significantly, and they are now employed to carry out sophisticated applications.

FPGAs are rapidly being used for recent cryptographic applications, which range from low power low cost implementations to fully pipelined parallel designs. Because FPGAs take less time to market than their Application Specific Integrated Circuit (ASIC) counterparts (Kuon and Rose, 2006) [5], their utilisation has increased.

Additionally, compared to ASIC design, the development process for FPGAs is far more efficient and less expensive. Additionally, unlike ASICs, which lack this feature, FPGAs' programmable nature allows the designer to alter the algorithm that was first developed. The device can be configured at runtime using this capability. Additionally, it can be used to optimise the algorithm for a predetermined set of requirements and repair the shortcomings in designs that have already been implemented (Dandalis and Prasanna, 2004) [6].

II. A Summary of AES

AES is a block cypher technique that supports block sizes of 128 bits and key sizes of 128, 192, and 256 bits. It encrypts data on a per-block basis (block matrix). Substitution Permutation Networks (SPN) serve as its foundation. Shannon created SPN to defend cyphers from attacks using statistical cryptanalysis. Block cyphers use SPN, which is a sequential set of mathematical operations (substitution and linear transformation). To better understand how AES functions, let's define a few crucial terms [7]–[8].

a) Round

In order to achieve a higher output cypher text, a round is a transformation procedure employed throughout the encryption process that comprises functions like substitution, transposition, and mixing. As seen in Table I [8], each key size is linked to the number of rounds that offer a very safe system.

Table I Structure of AES [8]

AES Type + Key Size	Data Block Size	Matrix Block	Number of Blocks
AES-128	128	4*4	10
AES-192	128	4*6	12
AES-256	128	4*8	14

b) The Block Cypher

Data is encrypted using a block cypher, which use the same key for both encryption and decryption. In fact, even if the same message is encrypted repeatedly, the block cypher method's goal is to prevent the cypher text from becoming similar during the encryption process. AES 128 is represented in a block of matrix 4 × 4 in Table II.

Table II Representation of AES 128 in block of matrix 4 × 4 [8]

a)

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Ten rounds of substitution and permutation are carried out by the cypher module to convert the input data into ciphered data when using a 128-bit key. The cypher module uses Sub Byte, Shift Row, Mix Column, and Add Round Key operations for the first nine encryption rounds. For the final round, the Mix column operation is omitted to finish the encryption process. The key expansion module determines the key utilised in each round by the aforementioned procedures based on the initial key. The term "state" refers to the 4 × 4-byte array that is frequently used to represent input data in AES.

Each round of the encryption process consists of four steps:
• Sub Bytes: This function individually applies a non-linear change to every byte in the input state. Each byte of the state is changed by replacing it with a value from the substitution box, also known as the S-box. Each of the sixteen parallel S-boxes has eight inputs and eight outputs. The AES algorithm's single nonlinear transformation is the S-box operation. It can also be utilised for the decryption process and is an invertible operation. By combining two transformations, the S-box

is constructed. In the first transformation, an all-zero bit input is mapped to itself by taking the multiplicative inverse in the finite field GF(28).

The affine transformation over GF(2) is carried out in the second section. In GF(2), the input is an 8-bit vector that is applied to 8-bit vector C after being multiplied by a constant 8 by 8-bit matrix M. Below are the constant matrices M and C [7].

$$M=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix} \qquad S'=\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}$$

• Shift Rows: This function moves the state's final three rows in a circular fashion, byte by byte. This function rotates the second, third, and fourth rows by one, two, and three bytes, respectively, while leaving the first row unaltered. The AES algorithm's diffusion property is provided by this rotation.

• Mix columns: This function treats each of the four columns of states as a four-term polynomial and performs independent operations on each of them. With a fixed polynomial a (x) provided by [7], the columns are multiplied modulo X4 + 1 and treated as polynomials over GF(28).

$$a(x) = \{03\}\, x^3 + \{01\}\, x^2 + \{01\}\, x + \{02\} \tag{1}$$

Above function can also be written as matrix multiplication as [8]

$$S'(x) = a(x).\, S(x) \tag{2}$$

$$\begin{bmatrix}S'0,i\\S'1,i\\S'2,i\\S'3,i\end{bmatrix} = \begin{bmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{bmatrix}\begin{bmatrix}S0,i\\S1,i\\S2,i\\S3,i\end{bmatrix} \quad \text{for} \quad 0 \le i \le 3$$

The last cypher function, Add Round Key, is used to combine important information with the data being worked on. The 16-byte state and 16-byte key, which are obtained by the key expansion method, are the inputs of this function. It produces a straightforward bit-wise XOR between the expanded key and the current round state [8].

$$S'0,i = (\{2\}.S\ 0,i) \oplus (\{3\}.\ S\ 1,i) \oplus (\{1\}.\ S\ 2,i) \oplus (\{1\}.\ S\ 3,i) \tag{3}$$

$$S'1,i = (\{1\}.\ S\ 0,i) \oplus (\{2\}.\ S\ 1,i) \oplus (\{3\}.\ S\ 2,i) \oplus (\{1\}.\ S\ 3,i) \tag{4}$$

$$S'2,i = (\{1\}.\ S\ 0,i) \oplus (\{1\}.\ S\ 1,i) \oplus (\{2\}.\ S\ 2,i) \oplus (\{3\}.\ S\ 3,i) \tag{5}$$

$$S'3,i = (\{3\}.\ S\ 0,i) \oplus (\{1\}.\ S\ 1,i) \oplus (\{1\}.\ S\ 2,i) \oplus (\{2\}.\ S\ 3,i) \tag{6}$$

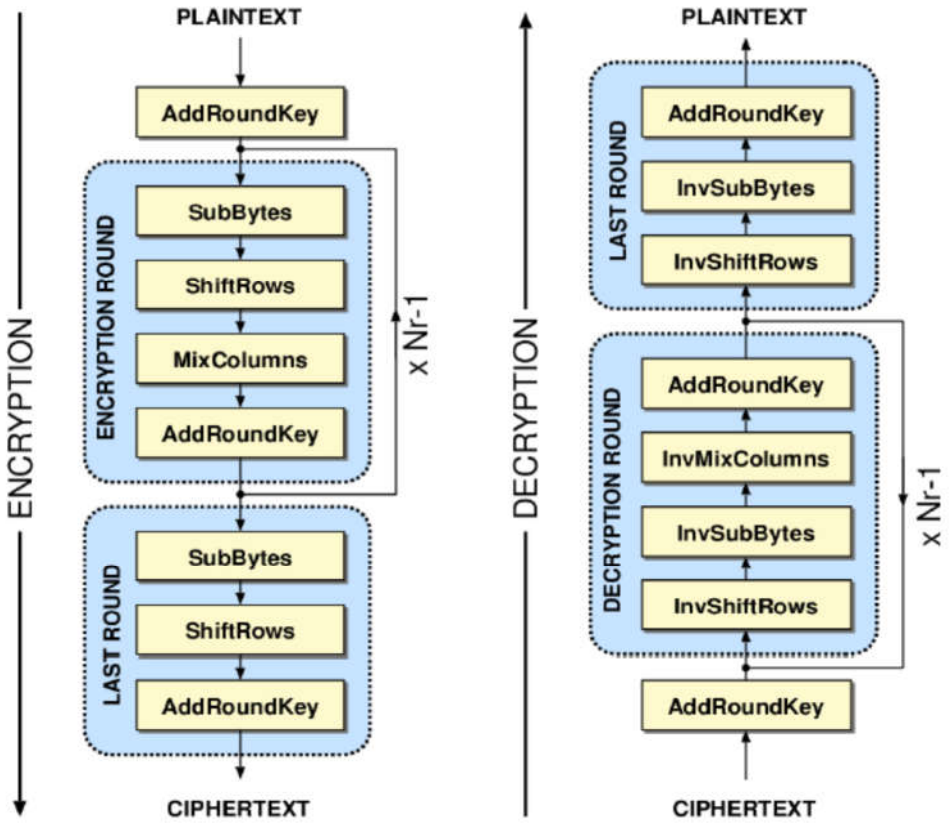

Figure 1: Encryption and Decryption process [8]

The steps in the encryption and decryption processes, which are inverse functions of one another, are shown in Figure 1. Below is an explanation of how the AES encryption method operates in detail.

• Layer of Key Addition (Add Round Key)

Figure 2 illustrates how a 128-bit round key, or sub key, that was produced from the main key in the key schedule is XORed with the State [9].

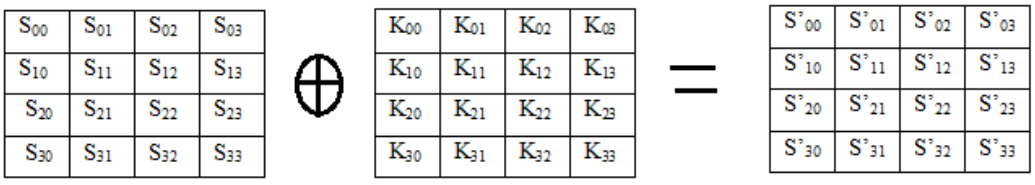

Figure 2 Key Addition Process [9]

The Layer of Confusion

By switching between plaintext and other content in the content state table, it creates confusion. Byte Substitution layer (S-Box): Lookup tables known as S-box (figure 3) [9] with unique mathematical properties are used to nonlinearly change each state element. As a result, the unencrypted data becomes confused [10].

X \ Y	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	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	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	6	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	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	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

Figure 3 S-Box [11]

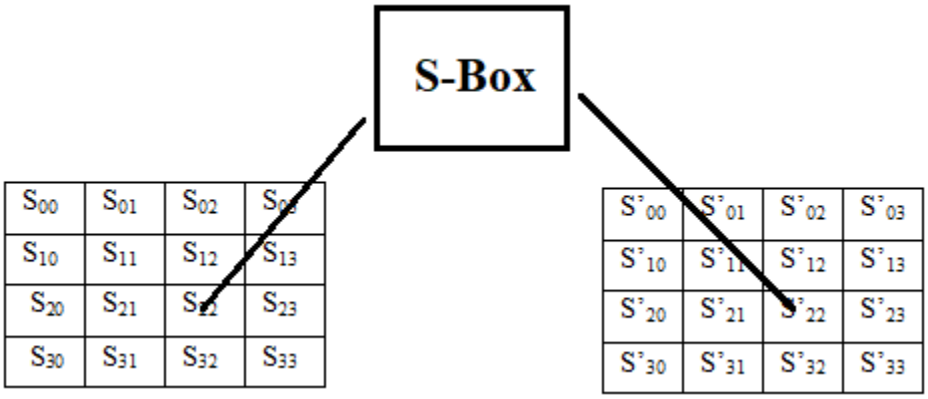

Figure 4 Byte Substitution Process

The diffusion layer

Diffusion over all state bits is provided. It is made up of two sublayers that both carry out linear operations: The technique for moving the rows (Figure 5) [9] of the output from the above layer is provided by the Shift Rows layer.

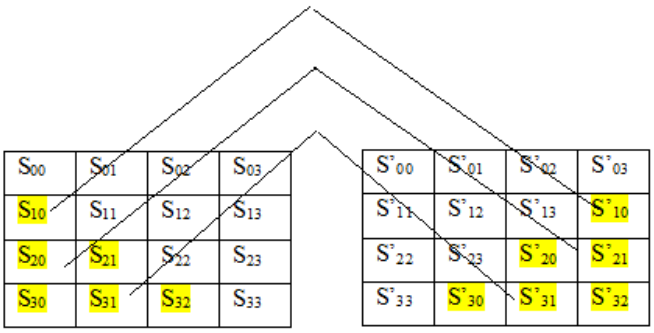

Figure 5 Mix Column Process

b. Layer of Mix Columns:

In this matrix operation, a fixed 4×4 matrix is multiplied by each 4-byte column, which is treated as a vector. There are constant entries in the matrix (Figure 6) [9]

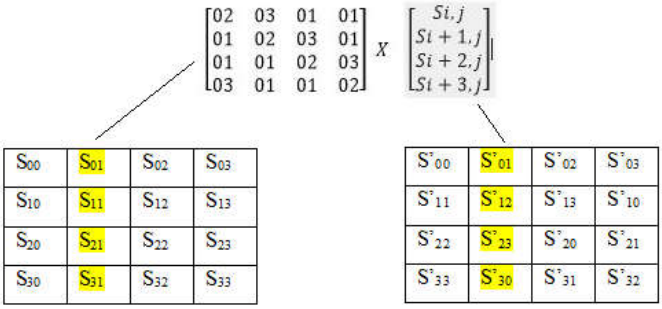

Figure 6 Mix Column Process

Figure 1 depicts the AES's entire procedure. The AES is stronger since the last round does not include the mix column step. The encryption process that performs inverse byte substitution, inverse shift row, and inverse mix column is reversed in the decryption process.
Each iteration of the decryption procedure consists of four phases [8]:

Inverse mix columns; inverse shift rows; inverse substitute bytes; and add round keys.

Figure 1 illustrates the algorithm's input plaintext, which is 128 bits in size; the key, which is 128 bits long, 192 bits long, or 256 bits long; and the cypher text, which is 128 bits long. The size of the key, which ranges from 10 to 14, determines how many rounds there are. To recover the original plaintext while maintaining the same key as the encryption process, the ciphertext generated from the encryption algorithm is fed into the decryption algorithm.

III. Associated Work

One of the key areas that draws academics is the implementation of the AES algorithm in hardware and software. Numerous studies on the AES algorithm have been released in recent years in an effort to simplify the algorithm and evaluate the effectiveness of the widely used encryption techniques for data encryption and decryption.Five distinct methods are used in [7] to implement AES on FPGA. These methods are based on the efficient use of the target device's resources to implement AES on FPGA in an optimised manner. The nature of the experimental results varies greatly. They range from the smallest (fit for applications requiring a lot of space) to the fastest (fit for applications requiring a lot

of performance). On a Spartan-6 device, the technique's effective use of resources results in a frequency of 886.64 MHz and a throughput of 113.5 Gb/s with a moderate resource consumption. Additionally, a comparison of the suggested method with previous research reveals that it has a 32% higher frequency despite using 2.63_ more slice LUTs, 8.33_ fewer slice registers, and 12.59_ fewer LUT-FF pairings.

In order to simplify the AES algorithm's complex architecture when it is implemented on hardware like smartphones, PDAS, smart cards, etc., Lu et al. presented a new architecture in [12]. To create a flawless, working AES crypto-engine, this technique involved combining the AES encryption and decryption blocks. They did this by concentrating on a few key AES elements, particularly the (Inv)Sub Bytes and (Inv)Mix column module. A study on various secret key algorithms was carried out in [13] in order to determine which method could encrypt and decrypt data with the best performance. Four popular algorithms, including Blowfish, AES, DES, and 3DES, were used in the experiment. Two distinct platforms, namely P-II 266 MHz and P-4 2.4 GHz, were utilised to test these methods, and the contents and sizes of the encrypted input files were altered. The results show that AES performs better than 3DES and DES, and that Blowfish can offer the best performance when compared to other algorithms. The results also indicate that 3DES's throughput is one-third that of DES.

Researchers evaluate the performance of symmetric encryption methods in [14]. Six popular algorithms, including AES, DES, 3DES, RC2, Blowfish, and RC6, were used in the experiment. Various factors were chosen in order to compare various methods, such as (a) different data kinds, (b) different data block sizes, (c) different key sizes, (d) battery power consumption, and (e) varied encryption and decryption speeds.There was little difference in algorithm performance whether employing audio, video, text, or documents when the data types were based on hexadecimal encoding or 64 encoding. The results showed that when the packet size changed, Blowfish outperformed other algorithms, with RC6 coming in second. However, when compared against the 3DES algorithm, researchers discovered that DES performs better. Out of all the algorithms, RC2 performed the poorest in terms of time consumption. In contrast, AES performs better than the three widely used algorithms, RC2, DES, and 3DES. Nonetheless, it is evident from the data that battery and time usage rose as key size increased.

Researchers assess how well three algorithms—AES, DES, and RSA—perform in encrypting text files for three parameters—calculation time, memory utilisation, and output bytes—in [15]. To determine whether approach takes longer to encrypt a text file, the encryption time to convert plaintext to ciphertext was calculated and compared. The results show that, in comparison to other techniques, RSA takes longer. Compared to AES and DES algorithms, RSA requires more memory for second parameters. Lastly, each algorithm's output byte was taken into account. The output byte level of RSA is low, while that of DES and AES is the same.

Recently, there has been a lot of effort done on the effective implementation of AES on FPGA. The sequential implementation of the AES algorithm on FPGA, for instance, is presented by the authors in [16] and produces a design that is ideal for small embedded applications. The invention of low power FPGA-based encryption algorithms is the main emphasis of implementation in [17]. These schemes aim to maximise power results without sacrificing the design's throughput. Three implementation plans are shown, and their areas and power consumption rates are contrasted. In a similar vein, the authors of [18] offer an effective FPGA version of the AES algorithm that produces high throughput and is ideal for applications needing high performance and speed.

In addition, writers in [19]–[21] investigate loop unrolling, pipelining, and subpipelining strategies to boost the frequency and throughput of AES implementation on FPGA. The study in [19] obtained a maximum throughput of 73.737 Gb/s and a maximum frequency of 576.07 MHz by using inner pipeline at two, three, or four stages on Xilinx Virtex-5 FPGA [22]. [23] uses a folded parallel architecture to achieve fast throughput. To increase space utilisation while preserving high throughput, the folding idea is applied. On a Xilinx Virtex-6 chip, the authors have reached a maximum frequency of 505.5 MHz. Recent research [24] shows a high accelerate the Xilinx Virtex-5 device's hardware version of the AES algorithm. They have attained a theoretical maximum frequency of 671.524 MHz and a throughput of 86 Gb/s.

IV. The Suggested Model

We suggest a modified AES system with a Double S-Box to increase the security of the encrypted data since data security is crucial given the increasing threat of cyberattacks. When an S-box is doubly permuted, a double S-box is produced. The non-linearity provided by the encryption technique does not change, even while the computation time increases. By entering plaintext and keys in several forms, such as "Text" and "Hex," the accuracy of the suggested

algorithm is confirmed, and encryption and decryption scans are performed to confirm the security the algorithm provides.

Plaintext: b'thisisplaintexts'
Key: b'mysecretpassword'
Ciphertext: b'\xeb\x06\\\xb1\x10\xc2\xeb\xf8\xff\xb1\x08\x8f\x16P'
Decrypted Text: b'thisisplaintexts'

Figure 7 DS-Box MAES used to encrypt and decrypt Text input

Plaintext: FF433B87D5D2EC551CFCF4C06FB3172A
Key: 75B67BE703FB2628D0D702519A94AC38
Ciphertext: AAC63C78E0A2315C41C8FE232E546693
Decrypted Text: FF433B87D5D2EC551CFCF4C06FB3172A

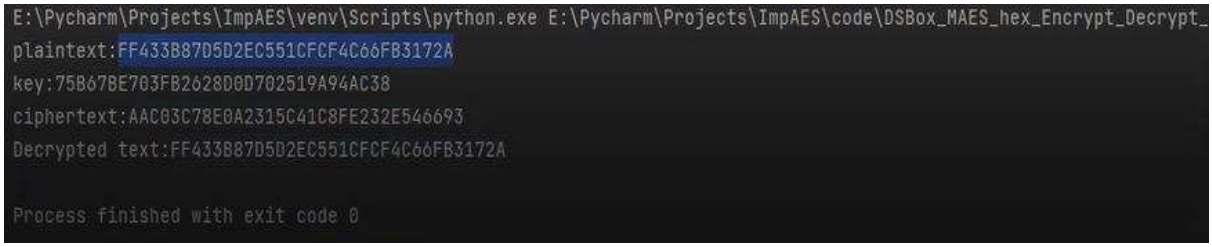

Figure 8 DS-Box MAES used to encrypt and decrypt Hex input

A PDF file with a size of 37.81KB was used as input, and the encryption and decryption durations were computed to determine how long DS-Box Modified AES would take. The data was encrypted at a rate of 0.0109 seconds per KB using the DS-Box modified AES technique, whereas the decryption time was 0.0326 seconds per KB of data (Figure 8).

--- MODIFIED AES ---

Input File: input.pdf
Size: 37.81 KB

Encryption Time: 0.4121 seconds
Encryption Time per KB: 0.0109 seconds

Decryption Time: 1.2315 seconds
Decryption Time per KB: 0.0326 seconds

Process finished with exit code 0

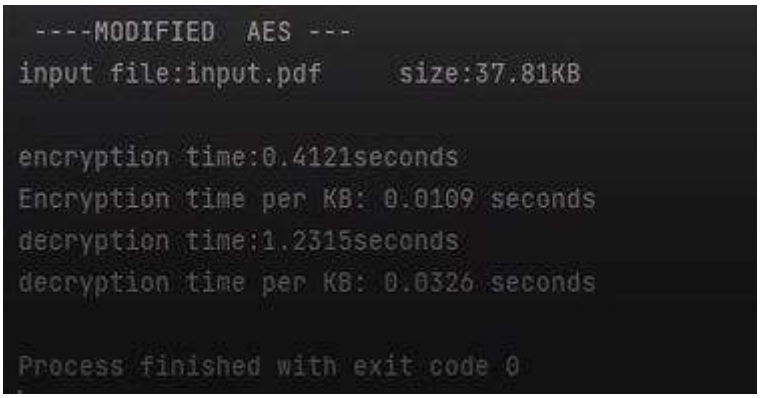

Figure 9 DS-Box MAES encryption and decryption time

Any encryption algorithm's performance is assessed using the "Monte Carlo" test. Numerous applications, such as risk analysis, numerical integration, and optimisation, use Monte Carlo methods. When examining the risks and uncertainties in an encryption algorithm, risk analysis is crucial. The Monte Carlo Test. In project management and financial investment, tests are also used to analyse risk. According to the results, DS-Box Modified AES is 0.90 times quicker than original AES and passed the Monte Carlo test (Figure 9).

Monte Carlo Test Successful!!

Original AES Performance: 0.14 seconds

Modified AES Performance: 0.16 seconds

Modified AES is 0.90 times faster than Original AES

Side Channel (Power Analysis) Test Successful!!

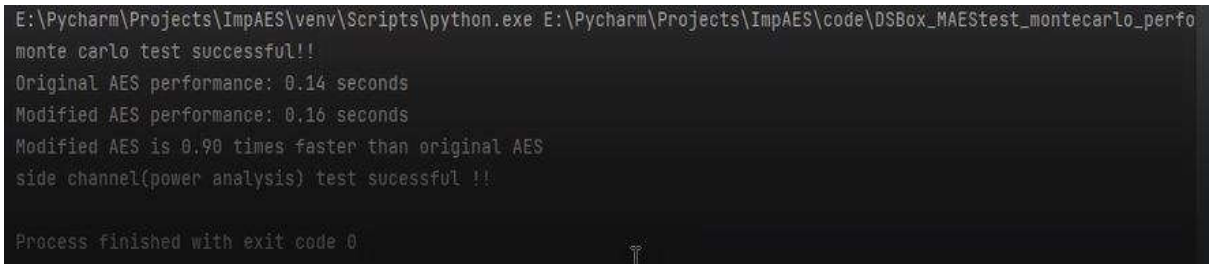

Figure 10 DS-Box MAES Monte Carlo Test Performance

The modified AES was exposed to the avalanche effect in order to assess its strength. In encryption algorithms, the avalanche effect is used to evaluate the algorithm's robustness to a 1-bit change in the plaintext or a key change (Figure 10, 11).

------ Avalanche Effect for 1-bit Change in Plaintext ------

Plaintext: FF433B87D5D2EC551CFCF4C06FB3172A

Key: 75B67BE703FB2628D0D702519A94AC38

OAES Cipher: DB5A2A63AA2187A502121F10AB3CEBC9
OAES Changed Plaintext: 7F433B87D5D2EC551CFCF4C06FB3172A
OAES Cipher due to changed plaintext: BA8EFFD50BF901A899E02679FEAC345
OAES Bits flipped for 1-bit change in plaintext: 62

OAES Avalanche for Plaintext in %: 48.4375

MAES Cipher: AAC03C78E0A2315C41C8FE232E546693
MAES Changed Plaintext: 7F433B87D5D2EC551CFCF4C06FB3172A
MAES Cipher due to changed plaintext: 0B63165BF7A22D31537289B263458140
MAES Bits flipped for 1-bit change in plaintext: 70

MAES Avalanche for Plaintext in %: 54.6875

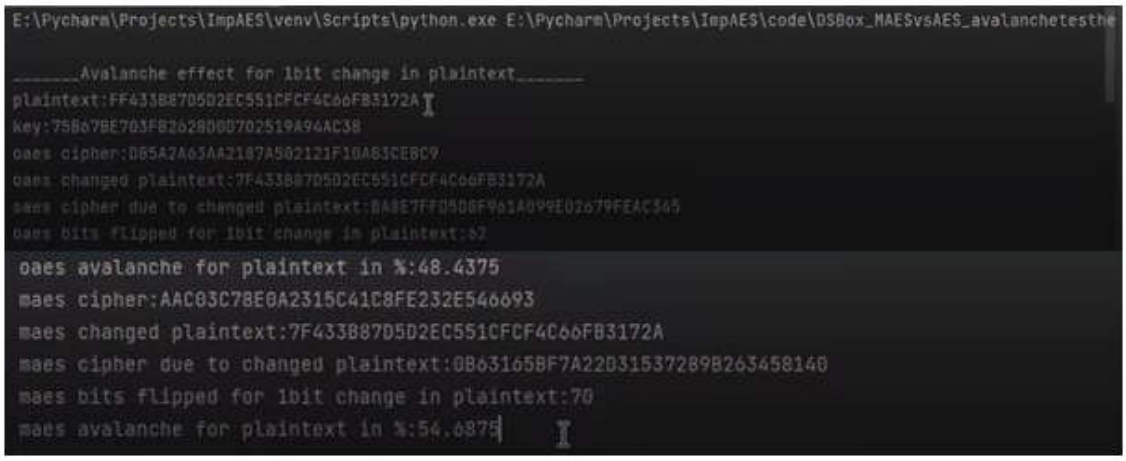

Figure 11 Avalanche effect for 1-bit change in plaintext

While 62 bits of the original AES flipped with an avalanche of 48.4375% for a 1-bit change in plaintext, Figure 10 demonstrates that the identical change flips 70 bits in MAES with an avalanche of 54.6875%.

------ Avalanche Effect for 1-bit Change in Key ------

Plaintext: FF433B87D5D2EC551CFCF4C06FB3172A

Key: 75B67BE703FB2628D0D702519A94AC38

OAES Cipher: DB5A2A63AA2187A502121F10AB3CEBC9

OAES Changed Key: F5B67BE703FB2628D0D702519A94AC38

OAES Cipher due to Changed Key: 167556FE1F539E00713BDD1840E77AEC

OAES Bits Flipped for 1-bit Change in Key: 62

OAES Avalanche for Key (%): 48.4375

MAES Cipher: AAC03C78E0A2315C41C8FE232E546693

MAES Changed Key: F5B67BE703FB2628D0D702519A94AC38

MAES Cipher due to Changed Key: AAD402A187B2CC87CA015C34F689BE1E

MAES Bits Flipped for 1-bit Change in Key: 64

MAES Avalanche for Key (%): 50.0

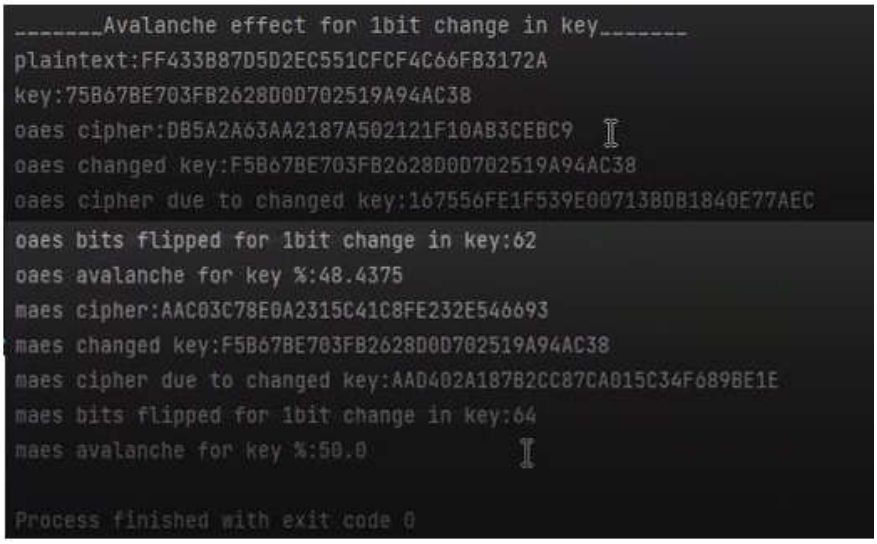

Figure 12 Avalanche effect for 1-bit change in key

Figure 11 illustrates how a 1-bit change in plaintext causes 62 bits of the original AES to flip, resulting in a 48.4375% avalanche, but the same change causes 64 bits in MAES to flip, resulting in a 50% avalanche. The avalanche effect generated by AES and DS-Box MAES for varying numbers of bits flipped is contrasted in Figure 12. The strength of DS-Box is demonstrated by the greater avalanche effect it produces.

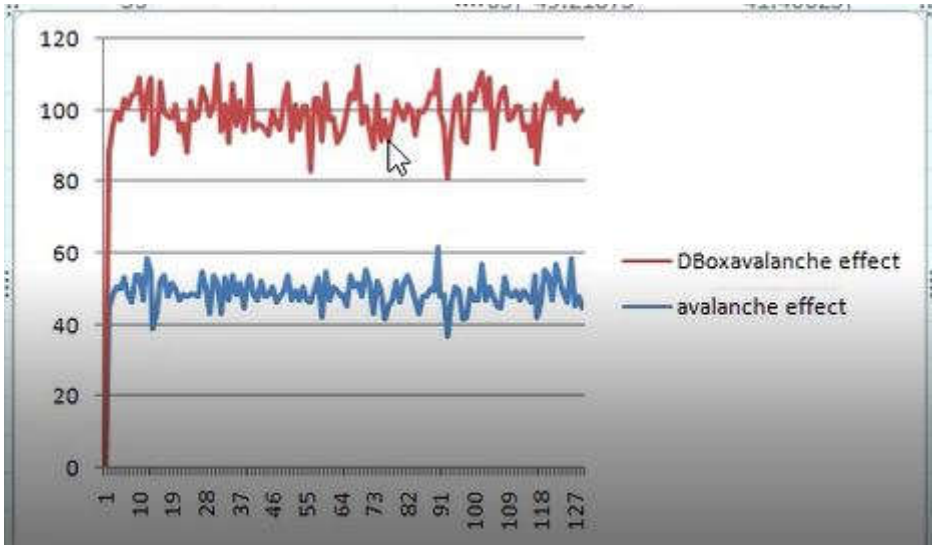

Figure 13 Comparing Avalanche effect for AES and DS-Box MAES

Another test involved passing blocks of varying sizes, from 0 to 5000 blocks, to both the original and modified AES. The encryption and decryption timings for each block pass were noted (Figure 13). Because Double S-Box permutes the data twice, it takes a little longer to encrypt and decrypt than AES, but this is negligible given how much additional security it provides.

Table:1 Comparing Encryption and Decryption time for AES and DS-Box MAES for various block sizes

Sr.No.	Block Size	OAES Encryption Time	OAES Decryption Time	MAES Encryption Time	MAES Decryption Time
1	200	0.074066877	0.141127825	0.08009984	0.147168398
2	400	0.145131826	0.28108263	0.155141115	0.291301251
3	600	0.210200548	0.416883707	0.234260559	0.435949802
4	800	0.285706997	0.554652452	0.309889078	0.583340406
5	1000	0.354333162	0.685023785	0392289877	0.727679729
6	1200	0.425720453	0.823498964	0.464498758	0.868348598
7	1400	0.490194559	0.962930202	0.539716244	1.012172222
8	1600	0.559015989	1.095996857	0.618071318	1.151070595
9	1800	0.631447315	1.24209404	0.695967436	1.296185017
10	2000	0.702323198	1.376744747	0.777683735	1.445584536
11	2200	0.777506113	1.514921427	0.852444649	1.588657379
12	2400	0.845208645	1.654974937	0.928346872	1.740486383
13	2600	0.912753344	1.788946867	1.002944946	1.880741358
14	2800	0.984912634	1.929254532	1.083525658	2.023365259
15	3000	1.063691378	2.128425121	1.163057327	2.177926541
16	3200	1.136487007	2.223447561	1.247174025	2.343698978
17	3400	1.194372177	2.349601507	1.318815947	2.509010077
18	3600	1.270814896	2.466019631	1.397389651	2.622466803
19	3800	1.331059933	2.595863342	1.461258173	2.732951881
20	4000	1.401716948	2.740289688	1.532720089	2.881828547
21	4200	1.468021154	2.873097658	1.610837461	3.022615911

22	4400	1.540571213	3.008774757	1.693687677	3.176176311
23	4600	1.635286808	3.205151321	1.765437841	3.323390484
24	4800	1.710268736	3.309458733	1.889114141	3.523740531
25	5000	1.752941847	3.451825381	1.929100037	3.599351406

V. Conclusion

Tonnes of digital data are transferred between different users on the internet almost every day. A portion of the data is sensitive and must be shielded from hackers. Algorithms for encryption are essential for shielding original data from unwanted access. There are several different types of algorithms for encrypting data. One of the most effective algorithms is the Advanced Encryption Standard (AES), which is widely used and supported in both software and hardware. With a 128-bit block cypher, this technique offers the versatility of being employed with key sizes of 128, 192, and 256 bits.

This study focusses on several key aspects of the AES algorithm. According to the findings of earlier research, AES can offer significantly higher security than other algorithms like DES, 3DES, etc. Our approach offers a high level of security by utilising a modified AES with Double S-Box. The results show that DS-Box MAES is 0.90 times quicker than AES and passed the Monte Carlo test. Additionally, DS-Box MAES offers a greater avalanche effect than AES, making it more resilient to modifications in keys or plaintext.

References

[1]. Stallings, W., (2010). “Cryptography and Network Security”: Principles and Practice, fifth ed. Prentice Hall.

[2] Advanced Encryption Standard (AES), 2001.

[3] Saggese, G., Mazzeo, A., Mazzocca, N., Strollo, A., (2003). “An FPGA based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm”. 2778, 292–302.

[4] Elbirt, A., Yip, W., Chetwynd, B., Paar, C., (2001). “An FPGA-based performance evaluation of the AES bolck cipher candidate algorithm finalists”. IEEE Trans. Very Large Scale Integration. VLSI Syst. 9, 545–557.

[5] Kuon, I., Rose, J., (2006). Measuring the gap between FPGAs and ASICs. In: Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays. ACM, New York, NY, USA, pp. 21–30.

[6] Dandalis, A., Prasanna, V.K., (2004). “An adaptive cryptographic engine for internet protocol security architectures”. ACM Trans. Des. Autom. Electron. Syst. 9, 333–353.

[7] Umer Farooq, M. Faisal Aslam, (2017), “Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA”. Journal of King Saud University – Computer and Information Sciences Vol. 29, 295–302.

[8] Toa Bi Irie Guy-Cedric, Suchitra R. (2018), “A Comprehensive Study on AES 128 bit and AES 256 bit”. International Journal of Scientific Research in Computer Science and Engineering Vol 16, Issue 4, pp30-33.

[9] J. Daemen, and V. Rijmen, (2002),"The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag.

[10] N. Bhat, v. Shridhar, (2012), “FPGA Implementations of Advanced Encryption Standard: a survey,” International Journal of Advances In Engineering & Technology, vol. 3, issue 2, pp. 265-285, 2012.

[11] Jayant P. Bhoge, Dr. Prashant N. Chatur, (2014), “Avalanche Effect of AES Algorithm”, International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3101 - 3103
[12] Lu, C. C., & Tseng, S. Y. (2002). “Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter”. In Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on (pp. 277-285).
[13] Nadeem, H (2006). “A performance comparison of data encryption algorithms," *IEEE Information and Communication Technologies*, (pp. 84-89).
[14] Diaa, S., E, Hatem M. A. K., & Mohiy M. H. (2010, May) “Evaluating the Performance of Symmetric Encryption Algorithms”. International Journal of Network Security, Vol.10, No.3, (pp.213-219).
[15] Berent, A. (2013). “Advanced Encryption Standard by Example”. Document available at URL http://www. networkdls. com/Articles/AESbyExample. pdf (April 1 2007) Accessed: June.
[16] Rouvroy, G., Standaert, F.X., Quisquater, J.-J., Legat, J., (2004). “Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded application”s. In: Proceedings of the International Conference on Information Technology: Coding and Computing. ITCC 2004, vol. 2, pp. 583–587.
[17] Van Dyken, J., Delgado-Frias, J.G., (2010). “FPGA schemes for minimizing the power-throughput trade-off in executing the advanced encryption standard algorithm.” J. Syst. Archit. 56, 116–123.
[18] Hoang, T., Nguyen, V.L. (2012), “An efficient FPGA implementation of the advanced encryption standard algorithm’. In: 2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), pp. 1–4.
[19] Soliman, M.I., Abozaid, G.Y., (2011). “{FPGA} implementation and performance evaluation of a high throughput crypto coprocessor”. J. Parallel Distrib. Comput. 71, 1075–1084.
[20] Gielata, A., Russek, P., Wiatr, K., (2008). “AES hardware implementation in FPGA for algorithm acceleration purpose”. In: International Conference on Signals and Electronic Systems. ICSES ’08, pp. 137–140.
[21] Qu, S., Shou, G., Hu, Y., Guo, Z., Qian, Z., (2009). “High throughput, pipelined implementation of AES on FPGA”. In: International Symposium on Information Engineering and Electronic Commerce. IEEC ’09, pp. 542–545
[22] Xilinx, (2014a). Virtex-5, <http://www.xilinx.com/>.
[23] Rahimunnisa, K., Karthigaikumar, P., Rasheed, S., Jayakumar, J., SureshKumar, S., (2014). “FPGA implementation of AES algorithm for high throughput using folded parallel architecture”. Secur. Commun. Netw. 7, 2225–2236.
[24] Farashahi, R.R., Rashidi, B., Sayedi, S.M., (2014). “FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption Algorithm”. Microelectron. J. 45, 1014–1025.