AES Encryption: Study & Evaluation

Dalia Yehya
Rafik Hariri University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Mohamad Joudi
Rafik Hariri University
9 PUBLICATIONS **2** CITATIONS
SEE PROFILE

AES Encryption: Study & Evaluation

Ahmad-Loay Sousi

Dalia Yehya

Mohamad Joudi

CCEE552: Cryptography & Network Security

Presented to: Dr. Jad Nasreddine

Rafik Hariri University

November 2020

Table of Contents

Abstract

With the widespread internet communication and the vital need to send data securely, we started using ciphering & encryption. As our use of ciphering increased, so did the complexity of the algorithms used starting from the very basic substitution ciphering reaching today's most secure cipher, the AES. AES has superior features which allowed it to replace DES & almost every other known cipher, especially when it comes to security as AES is currently computationally unbreakable and will probably remain unbreakable unless a future quantum computer manages to reach the required computational ability. The encryption process involves Substitution of the bytes, shifting rows, mixing columns & adding round keys. The decryption process on the other hand involves adding round keys, inverse shifting rows, inverse byte substitution & inverse mixing columns. Current attacks have managed to target incomplete implementations of the algorithm without even coming close to breaking a Full-AES algorithm implemented correctly. In addition to security, the algorithm's efficiency, sustainability & simplicity leads to a high evaluation allowing it to be in such supreme position among all other ciphers.

I. Introduction

1. Overview

Internet communication plays a vital role in transferring tons and tons of data everyday to millions of users. Throughout the years, fear has risen regarding data sent through insecure channels that are subject to spying and manipulation by malicious users. To combat that, different security measures were taken to ensure that data or messages only reaches those authorized to receive them.

Cryptography was one of the main techniques deployed to secure data through the processes of Encryption & Decryption. Encryption involves encoding information for the sake of preventing intruders from reading the original intended message easily. This process involves turning "Plain Text" to unreadable "CipherText" using keys, substitutions, permutations or even a mix of the aforementioned. Decryption process, on the other hand, intends to convert ciphered text back to the original plain text without missing any character/letter from the original text. Performing both processes involved using mathematical calculations and certain algorithms.

The main concern of cryptography is providing confidentiality, integrity, nonrepudiation & authentication through encryption & decryption algorithms that can be classified into three types: symmetric cryptography, asymmetric cryptography & hashing. Symmetric cryptography relies on using the same key for both encryption & decryption. Asymmetric cryptography relies on using different keys (public & private) for the two processes. Hashing in its basic form does not require any key. The Advanced Encryption Algorithm (AES) is a symmetric algorithm which we will further discuss in detail in our research.

2. History

Who didn't play the game of "secret message" as a kid and substituted a letter by another? If anyone knew how this formula worked, the "secret message" would be exposed. Well, in cryptography, what we did as children was encrypting a message using a very simple mathematical algorithm. Encryption has been used to hide delicate data since ancient eras, but really arose on its own during the Twentieth Century. Man has spent thousands of years in the quest for strong encryption algorithms due to the increase of threats to computer and network security each passing day. We had DES, which is based on an algorithm developed by IBM, and it was considered unbreakable. But not for too long, brute-force attacks by the late 1990s were able to break it in a matter of several days. That was made possible due to the relatively small block size it had, key size and advances in computing power according to Moore's Law. As a result, DES was unable to take advantage of the rapid development in microprocessors that occurred in the last two decades of the 20th century. The achievement of breaking DES signaled the end of it. Yet, Triple DES is still acceptable for federal use until 2030. In January 1997, NIST announced a competition for the successor to DES and was made open, public, and the encryption algorithm was available for use royalty-free worldwide. Competitors had more than cryptographic strength to take into consideration since the ease of implementation and performance in software and hardware were basic requirements for winning the competition. Over the progression of three competitive rounds and fierce cryptanalysis by the world's leading specialists on encryption, NIST selected the winner, the Rijndael algorithm of Belgian cryptographers Joan Daemen and Vincent Rijmen in October 2000. AES at that time was approved for encryption of only non-classified governmental data. In 2003, AES was approved for use with Secret and Top secret classified information of the U.S. government.

3. Definition

The AES algorithm (also referred to as the Rijndael algorithm) is a symmetrical block cipher algorithm that uses 128,192, or 256 bit keys to transform a block of 128 bits message into a 128 bits of ciphertext which is the main reason why it is strong, secure and exponentially stronger than the DES that uses 56 bit key.

A substitution-permutation, or SP network, with several rounds is used by the AES algorithm to generate ciphertext. The key length used will determine the number of rounds. For example, in the encryption process the 128 bit keys consist of 10 rounds, 12 rounds for the 192 bit keys and 14 rounds for 256 bit keys. In each case, all the rounds are identical except for the last round we won't have the mix column step.

Instead of having a one line of bytes or bits like most ciphers, AES arranges them in a 4x4 grid.

Byte 00	Byte 04	Byte 08	Byte 12
Byte 01	Byte 05	Byte 09	Byte 13
Byte 02	Byte 06	Byte 10	Byte 14
Byte 03	Byte 07	Byte 11	Byte 15

4. AES VS DES

AES	DES
Based on a substitution-permutation principle.	Based on the Feistel cipher structure.
Key length is 128, 192, or 256 bits (more secure).	Key length is 56 bits (less secure).
Block size is 64 bits	Block size is 128 bits
The cipher type is symmetric block cipher	The cipher type is symmetric block cipher
The security is considered secure (virtually impenetrable)	The security is proven inadequate
The whole block of data is treated as a single matrix.	The data block is broken into two halves.
It consists of: 10 rounds for 128 bit keys. 12 rounds for the 192 bit keys. 14 rounds for 256 bit keys.	It consists of 16 rounds.
Subbytes, Shiftrows, Mix columns, Addroundkeys.	Expansion Permutation, Xor, S-box, P-box, Xor and Swap.
Faster	Slower

5. Advantages & Disadvantages of AES

a. Advantages

- It is the most strong security protocol, since it is applied in both hardware and software.
- It uses key sizes of greater length for encryption, such as128, 192 and 256 bits so it is more resistant to hacking. For example, Around 2128 attempts are required for 128 bits to break. This makes it very difficult to hack it, since it is a very secure protocol.
- For a wide range of applications, such as wireless communication, financial transactions, e-business, encrypted data storage, etc., it is the most common security protocol used.
- It is one of the world's most commonly used commercial and open source solutions.

b. Disadvantages

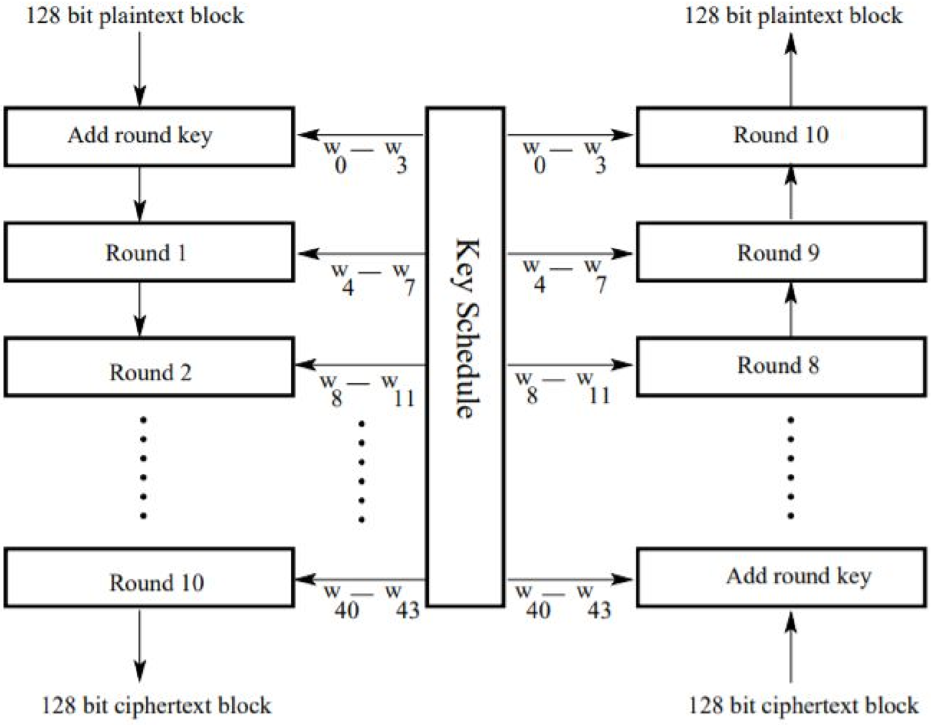
- It uses algebraic structures that are too simple and easy.
- All blocks are encrypted in the same way at all times.
- Difficult to implement it with software.
- In counter mode, AES is difficult to implement in software, taking into account both performance and security.

6. Overall Structure

The input state array is XORed with the first four words of the key schedule before any round-based processing for encryption can start. During decryption, the same thing happens, except that we now XOR the ciphertext state array with the last four key schedule words.
Each round consists of the following four steps for encryption: Substitute bytes, Shift rows, Mix columns and Add a round key. Lastly, we XOR the four words from the key schedule with the output of the previous three steps.

Each round consists of the following four steps for decryption: Inverse shift rows, Inverse substitute bytes, Add round key, and Inverse mix columns. Lastly, we XOR the four words from the key schedule with the output of the previous two steps.

The last step in encryption does not have the mix columns step, and the last step in decryption does not have the inverse mix columns step.



AES Encryption AES Decryption

II. Encryption Process

1. Preparation Steps

As mentioned before, each round consists of a substitution step, a row wise permutation step, a column-wise mixing step, and the addition of the round key. For encryption and decryption, the sequence in which these four steps are executed is different. While, overall, very similar steps are used in encryption and decryption, their execution is not equivalent and as previously stated, the order in which the steps are used is different. The first step of an AES encryption cipher is data substitution by using a substitution table. The second step is to change the data rows and the third step is to mix the columns. The last step is done using a different part of the encryption key on each column.

The first thing that happens under this encryption process is that your plaintext (which is the data you want to encrypt) is divided into blocks. The AES block size is 128 bit, so it divides the data into a 16 byte 4x4 column (16 x 8 = 128).
For example, if the message is “hello doctor jad”, the first block will look like the following:

h	o	c	
e		t	j
l	d	o	a
l	o	r	d

If there’s more of the message, it will be added to the next block. If solving by hand, it better to convert this message into hexadecimal which will look like the following:

h	e	l	l	o		d	o	c	t	o	r		j	a	d
68	65	6c	6c	6f	20	64	6f	63	74	6f	72	20	6a	61	64

Then we will do a key expansion which involves taking the initial key and using it for each round of the encryption process to come up with a set of other keys. With Rijndael's key schedule, these new 128-bit round keys are derived, which is basically an easy and fast way to generate new key ciphers. If the initial key was "two one nine two":

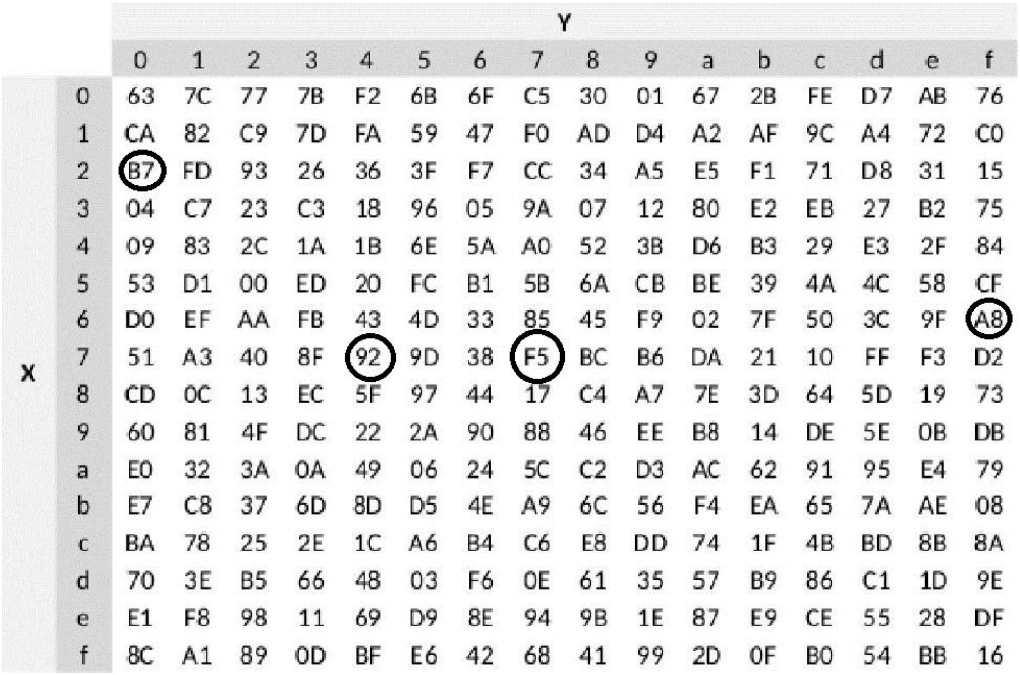
t	o	n	
w	n	i	t
o	e	n	w
		e	o

The hexadecimal representation of the above message is the following:

t	w	o		o	n	e		n	i	n	e		t	w	o
74	77	6f	20	6f	6e	65	20	6e	69	6e	65	20	74	77	6f

w[0] = (74, 77, 6f, 20)
w[1] = (6f, 6e, 65, 20)
w[2] = (6e, 69, 6e, 65)
w[3] = (20, 74, 77, 6f)
To find g(w[3]) we have to:

- Shift the bytes of w[3]: (74, 77, 6f, 20)
- Then we do a byte substitution (S-box) using the following table:



		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	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

We will get: (92, F5, A8, B7)

- We add a round constant (01, 00, 00, 00) which will give us g(w[3]) = (93, F5, A8, B7)

Then we use Rijndael’s key schedule to obtain or derive the new keys:

10010011 ⊕ 01110100	11110101 ⊕ 01110111	10101000 ⊕ 01101111	10110111 ⊕ 00100000
E7	82	C7	97

w[4] = w[0] ⊕ g(w[3]) = (E7, 82, C7, 97)

11100111 ⊕ 01101111	10000010 ⊕ 01101110	11000111 ⊕ 01100101	10010111 ⊕ 00100000
88	EC	A2	B7

w[5] = w[4] ⊕ w[1] = (88, EC, A2, B7)

10001000 ⊕ 01101110	10000010 ⊕ 01101001	10100010 ⊕ 01101110	10110111 ⊕ 01100101
F9	85	CC	D2

w[6] = w[5] ⊕ w[2] = (F9, 85, CC, D2)

11111001 ⊕ 00100000	10000101 ⊕ 01110100	11001100 ⊕ 01110111	11010010 ⊕ 01101111
D9	F1	BB	BD

w[7] = w[6] ⊕ w[3] = (D9, F1, BB, BD)

The first round key is: E7 82 C7 97 88 EC A2 B7 F9 85 CC D2 D9 F1 BB BD

In this stage we add round key, which means that our initial key is added to the block of our message since it is the first round:

68	6f	63	20
65	20	74	6a
6c	64	6f	61
6c	6f	72	64

+

74	6f	6e	20
77	6e	69	74
6f	65	6e	77
20	20	65	6f

=

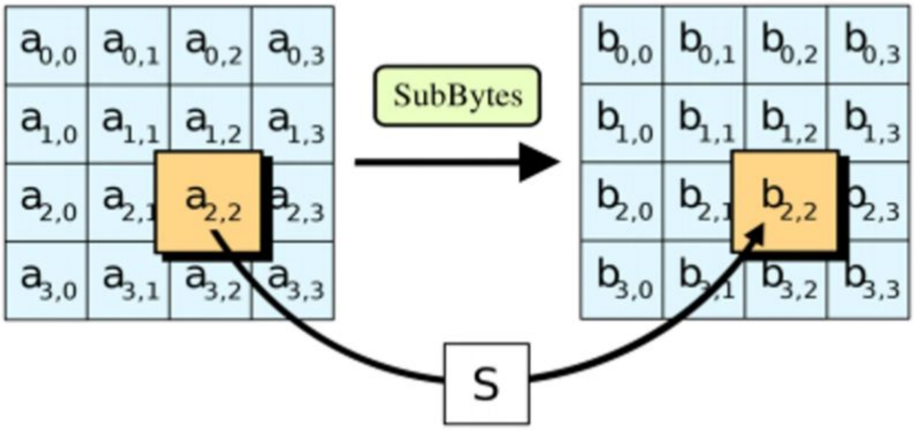
1c	00	0d	00
12	4e	1d	1e
03	01	01	16
4c	4f	17	0b

This is achieved using an XOR cipher, which is an algorithm for additive encryption. It may seem that it is hard to add these 2 tables, but in fact it is easy since it is achieved in binary or hexadecimal example, 68 ⊕ 74 = 1c.

2. Steps used in each round

a. Subbytes (Substitution of the bytes)

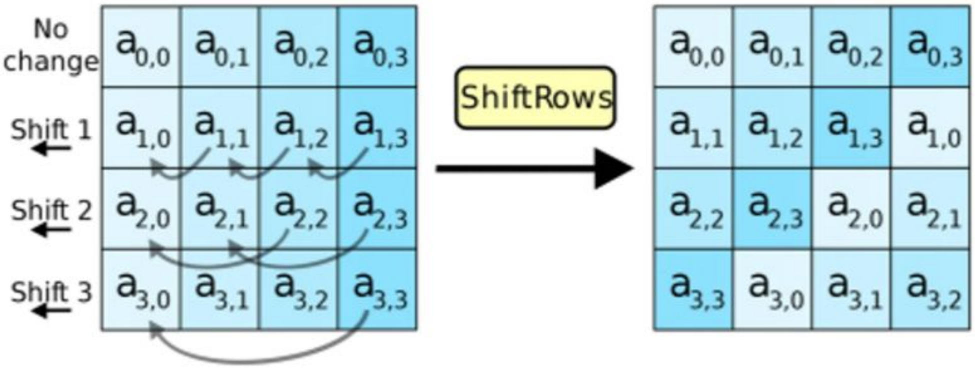
In the first step, we use a 16x16 lookup table to substitute the given byte in the input array. Using the notions of multiplicative inverses in GF(28) and bit scrambling, the entries in the lookup table are generated to remove the bit-level similarities inside each byte.



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
S
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

b. Shiftrows

This step is the permutation step. All the rows except the first one are shifted, the second row will be shifted one space to the left, the third row should be shifted two spaces to the left and the fourth row should be shifted three spaces to the left as seen below:

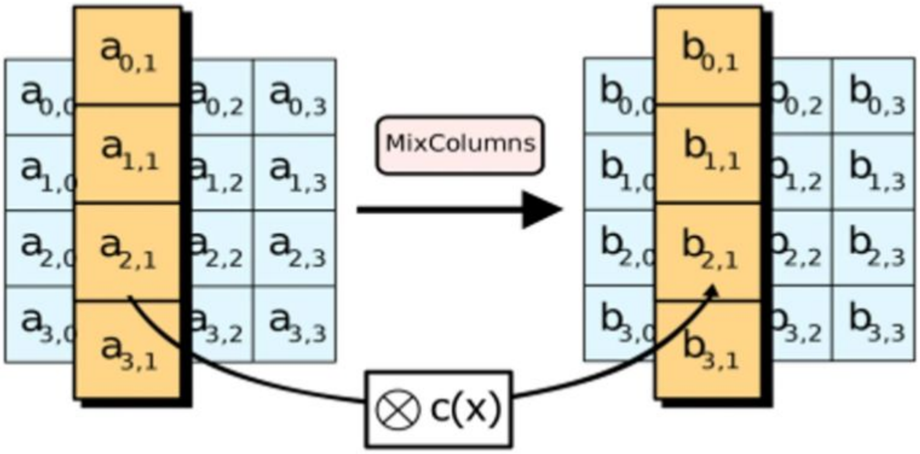


No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

After 10 rounds of processing, the shift-rows step along with the mix-column step makes every bit of the ciphertext to depend on every bit of the plaintext.

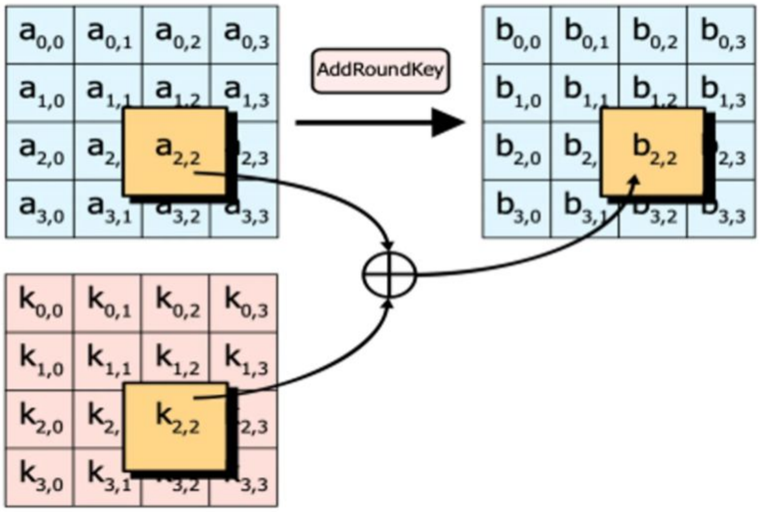
c. Mix columns

In the third step, the Hill cipher is used by changing the block's columns to mix up the message more or we can say that in order to further diffuse it, every column has a mathematical equation applied to it.


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

d. Addroundkeys

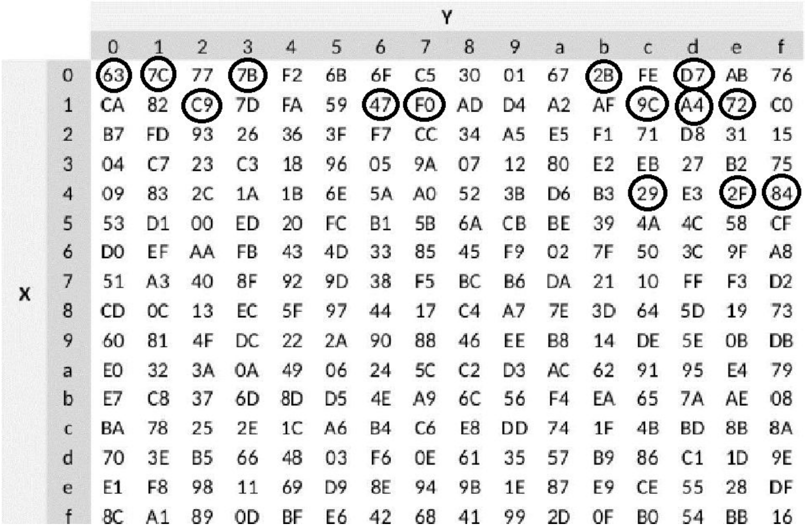
In the final step, the result of the mixed columns is XORED with the first round key that was derived from the start using the initial key and Rijndael’s key schedule.


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

If we want to implement the above steps on the example in this report, round 0 should look like:

- Subbytes (Substitution of the bytes):

  we substitute each entry by corresponding entry in AES S-box:



										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	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
X	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

1c	00	0d	00
12	4e	1d	1e
03	01	01	16
4c	4f	17	0b

9c	63	D7	63
C9	2F	A4	72
7B	7C	7C	47
29	84	F0	2B

- Shiftrows

The four rows are shifted to the left by an offset of 0, 1, 2 and 3.

9c	63	D7	63
C9	2F	A4	72
7B	7C	7C	47
29	84	F0	2B

↓

9C	63	D7	63
2F	A4	72	C9
7C	47	7B	7C
2B	29	84	F0

- Mix columns

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

9C	63	D7	63
2F	A4	72	C9
7C	47	7B	7C
2B	29	84	F0

The result of 9C of (02 • 9C) ⊕ (03 • 2F) ⊕ (01 • 7C) ⊕ (01 • 2B):

.

02• 9C = 00000010 • 10011100 = 0100111000
03• 2F = 000000011 • 01100011 = 010001101
01• 7C = 0000001 • 1111100 = 01111100
01• 2B= 000001 • 101011=0101011

Then we xor 0100111000 ⊕ 010001101⊕ 01111100⊕ 0101011 = 01001101100.The result of 9C is 26C.

The resulting table is:

26C	129	D7	63
F8	148	156	C9
B9	47	F5	174
CC	29	84	E0

- Addroundkeys

  In this step, we will add like before the output matrix of the mix column step and the first round key.

  Then we will do all these steps 10 times until we get an encrypted message (last round does not include addroundkeys).

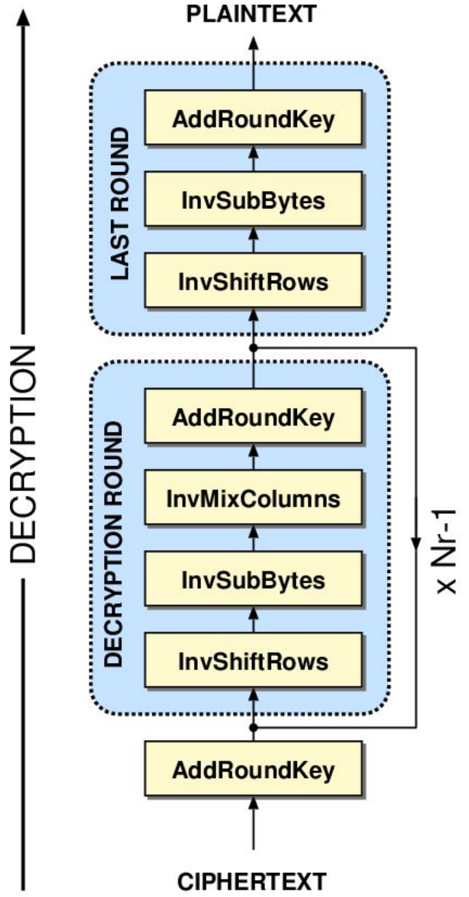
  The encrypted message of this example is:
  B1F2762902A0F09B4D3956C654C521668A5DDC7138E19205C5FAE9A6982E4B2D

III. Decryption Process

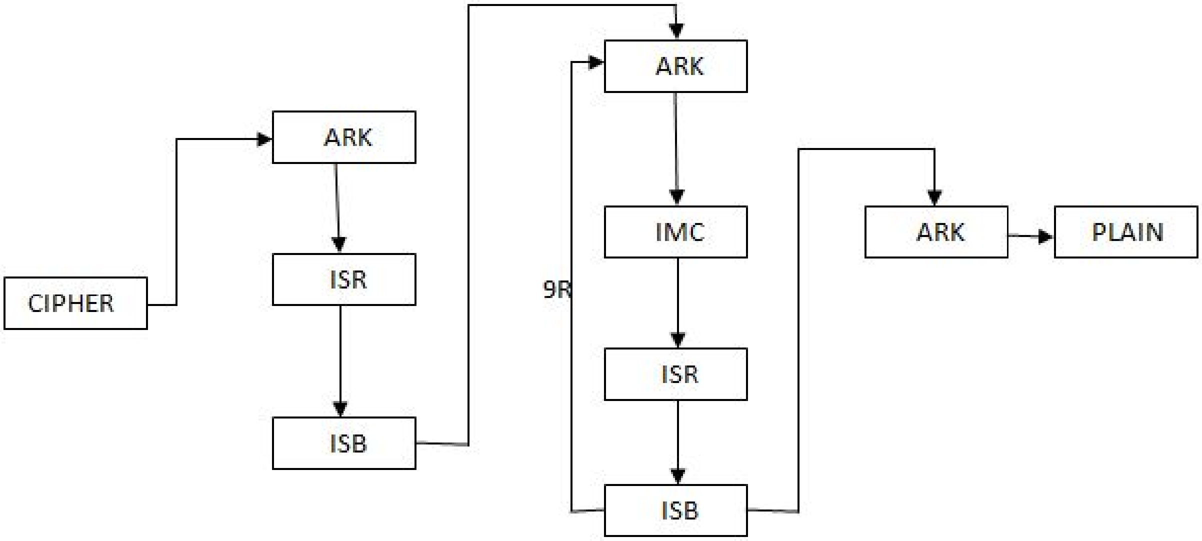
In AES, the decryption process depends on the key received from the sender. Both the sender and the receiver have the same key for encryption & decryption of data. This process is similar to the encryption process; however, in the reverse order.

Decryption starts with an initial round, followed by 9 iterations of an “inverse normal round” and ends with an “AddRoundKey”. An “inverse normal round” consists of the operations “AddRoundKey”, “InvMixColumns”, “InvShiftRows” & “InvSubBytes” respectively. The last round consists of the same operations of an “inverse normal round” except for the “InvMixColumns”.

The figure below demonstrates the process of decryption **for a different message**:



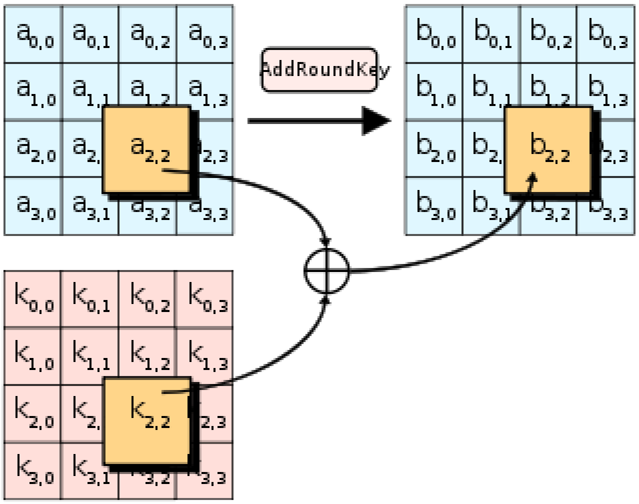
This figure also visually interprets the decryption process using a process map:



Map Keys: ARK: AddRoundKeys | ISR: Inverse Shift Rows | ISB: Inverse Sub Bytes | IMC: Inverse Mix Columns

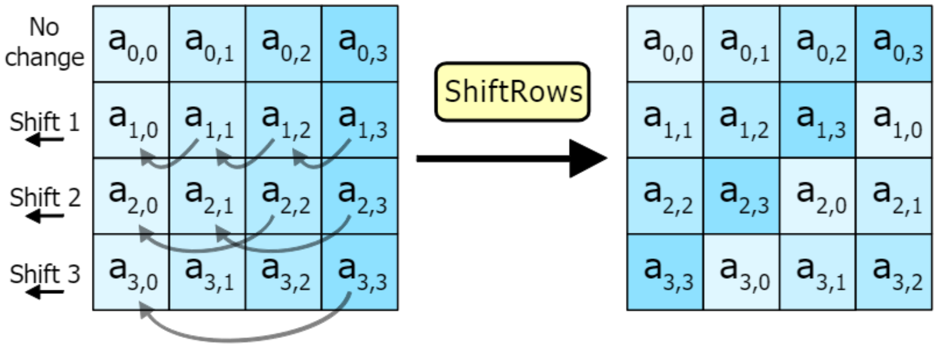
a. Addroundkeys

In the last round, we perform the AddRoundKeys as performed in the encryption algorithm in the previous section. We perform the XOR operation between the cipher text and the expanded key corresponding to the particular iteration.
Let's take the following figure with the diagrams on the left represent the cipher and the key values & the one on the right has the generated final value.



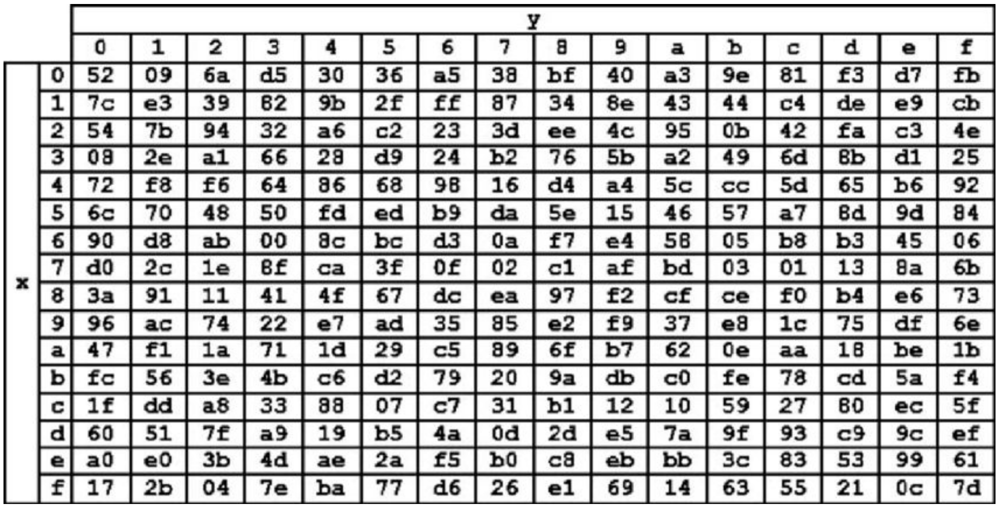
b. Inverse shift rows

In this step, we rotate each ith row by i elements to the right, as shown in the figure below:



c. Inverse byte substitution

In this step, we replace each entry in the matrix from the corresponding entry in the inverse S-Box, as shown in figure below



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

We now perform the following operations 9 times (9 iterations):

d. AddRoundKey in iteration stage (same as previous step)

e. Inverse mix columns in iteration stage

This operation is performed by the Rijndael cipher and it acts as the primary source of all the 10 rounds of diffusion in Rijndael. Each column is treated as a polynomial over Galois Field and multiplied by a fixed inverse polynomial as shown below

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

f. Inverse shift rows in iteration stage (same as previous step)

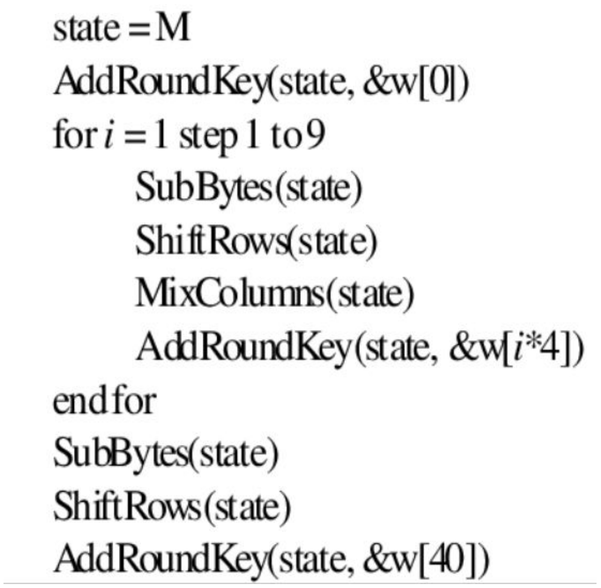
g. Inverse byte substitution in iteration stage (same as previous step)

IV. Implementation

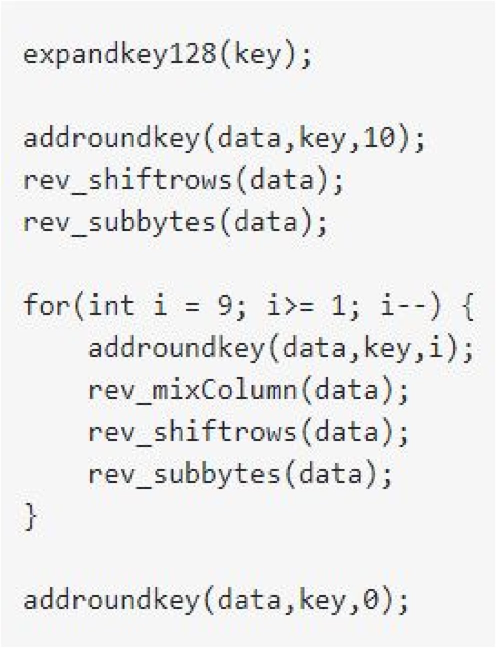
“AES” is one of the most powerful algorithms that are heavily used in various areas in computing. This algorithm is faster & more secure than “DES” & “3DES” algorithms for encrypting and decrypting data.

Furthermore, it is implemented in many “cryptography protocols” such as “Socket Security Layer (SSL)” & “Transport Security Layer (TSL)” to provide secure communications between a client and the server. In addition, “AES” is also present in most modern apps and devices that require guaranteed encryption such as “WhatsApp”, “Facebook Messenger”, “Intel & AMD” processor & “Cisco” hardware such as routers & switches. Moreover, “AES Crypt” package can be implemented using libraries in programming languages like “C++, C# /.NET, Java and JavaScript “ for the sake of securing & encrypting files from intruders.

Pseudocode:



```
state = M
AddRoundKey(state, &w[0])
for $i = 1$ step 1 to 9
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, &w[$i$*4])
endfor
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, &w[40])
```



```
expandkey128(key);

addroundkey(data,key,10);
rev_shiftrows(data);
rev_subbytes(data);

for(int i = 9; i>= 1; i--) {
    addroundkey(data,key,i);
    rev_mixColumn(data);
    rev_shiftrows(data);
    rev_subbytes(data);
}

addroundkey(data,key,0);
```

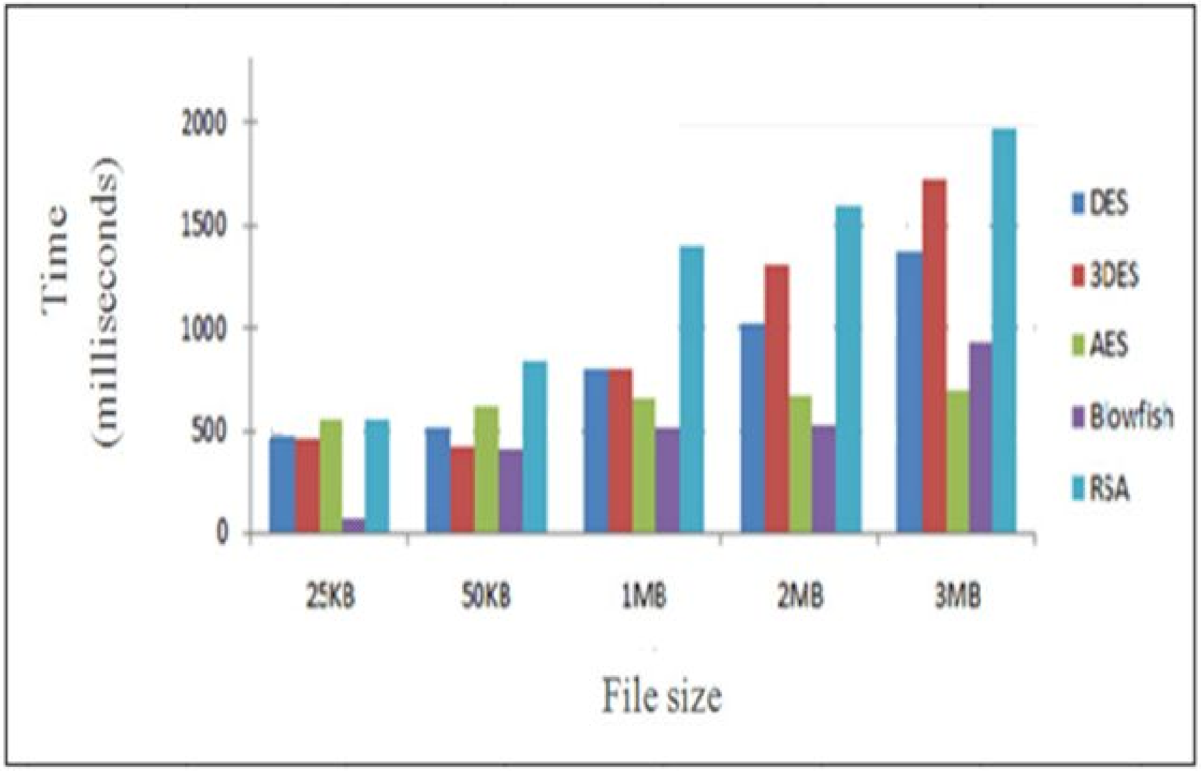
V. Evaluation

Strength Comparison (AES vs DES)

AES is a highly secure ciphering algorithm whether it’s 128-AES, 256-AES or 512-AES. While 512-AES is an overkill, the debate settles between the 128 & 256 bit variations. To make the picture clearer, both ciphers have never been broken so far as the possibilities (2^{128} & 2^{256}) available exceed the number of atoms in the universe thus bruteforce is not an option. One might think that 256-AES is twice as secure compared to 128-AES but in fact, it’s 340 billion-billion-billion-billion times harder. Even if we did manage to build a world-wide network of super-computers designed solely for the sake of testing combinations, it would take 100+ billion years to find the correct text. For comparison, the universe has only been around for 13.8 billion years. It is thus no surprise how AES (developed in 2000) replaced DES (developed in 1977) especially since DES was proven inadequate with a key-length of 56 bits & a block size of 64 bits.

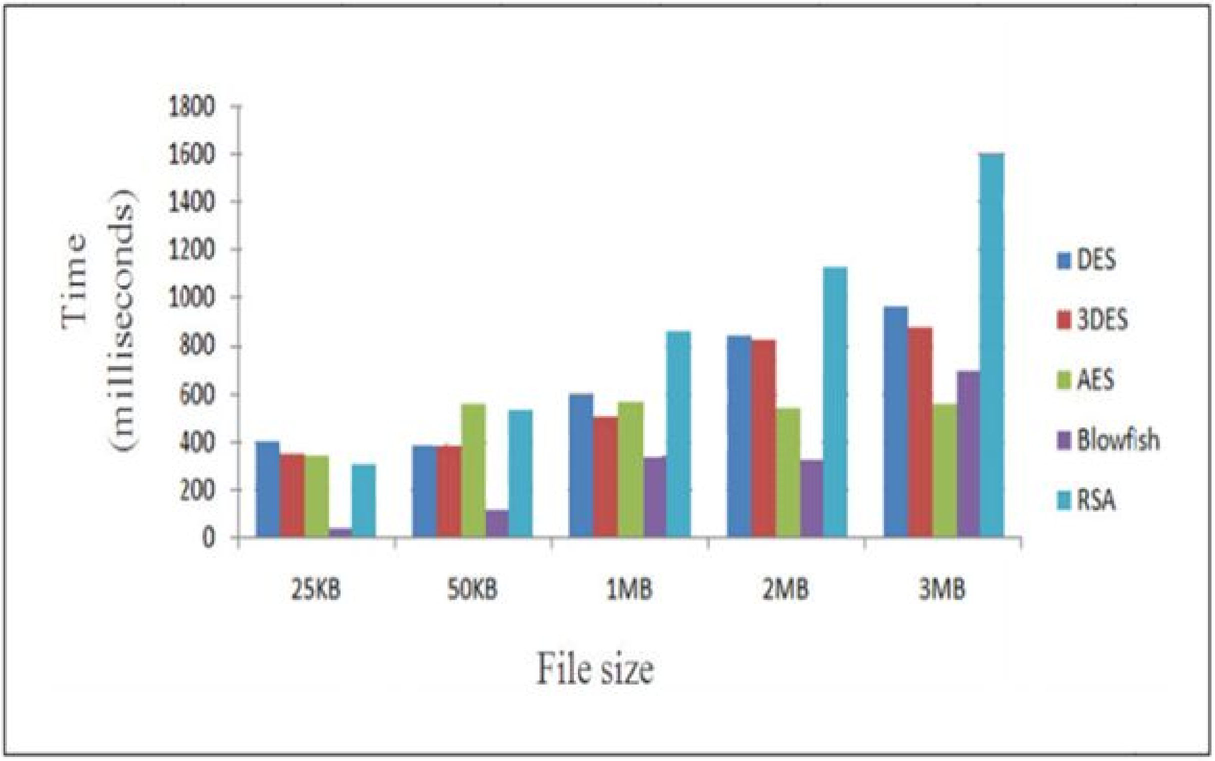
Speed Comparison (AES vs DES)

<u>During Encryption:</u>



We notice that DES is faster than AES which makes sense when comparing the 56 block size of DES compared to 128 block size of AES.

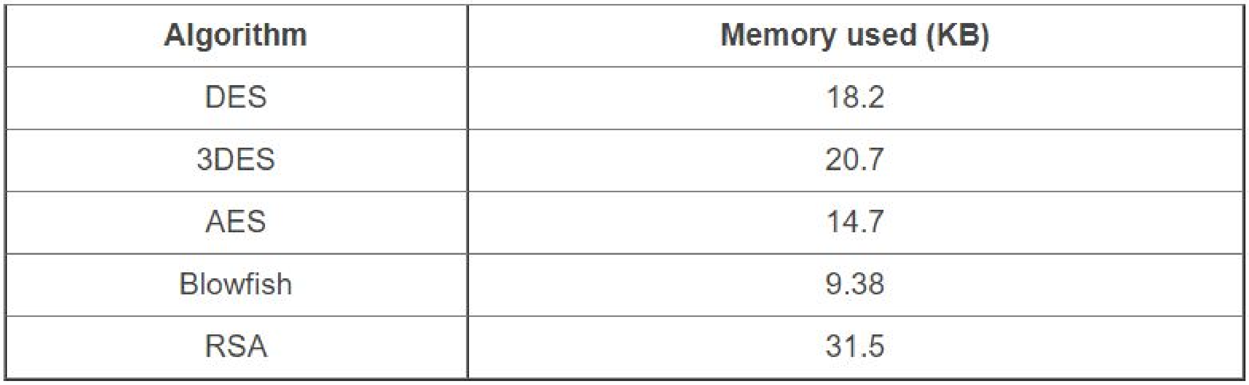
During Decryption:



Similarly, the decryption time is slightly higher for AES compared to DES.

Memory Requirement Comparison (AES vs DES)

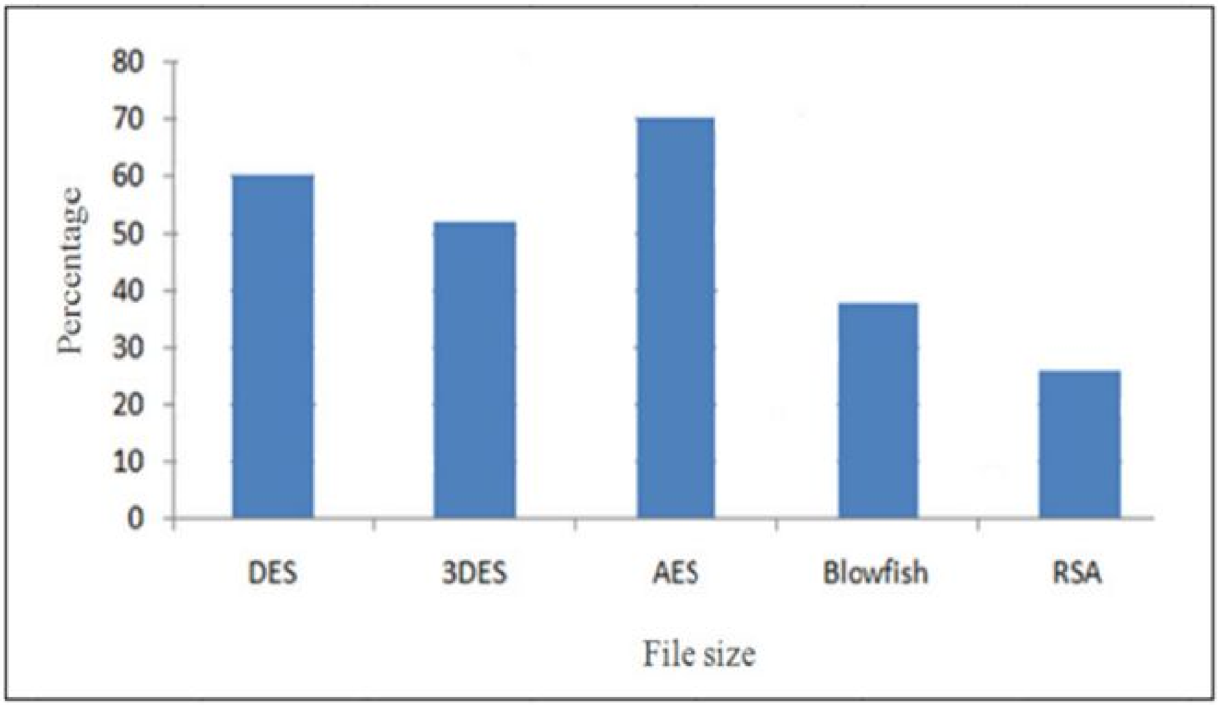
The AES algorithm uses less memory than DES algorithm due to the characteristics & implementation requirements of the algorithm, yet Blowfish Algorithm remains the most memory-friendly

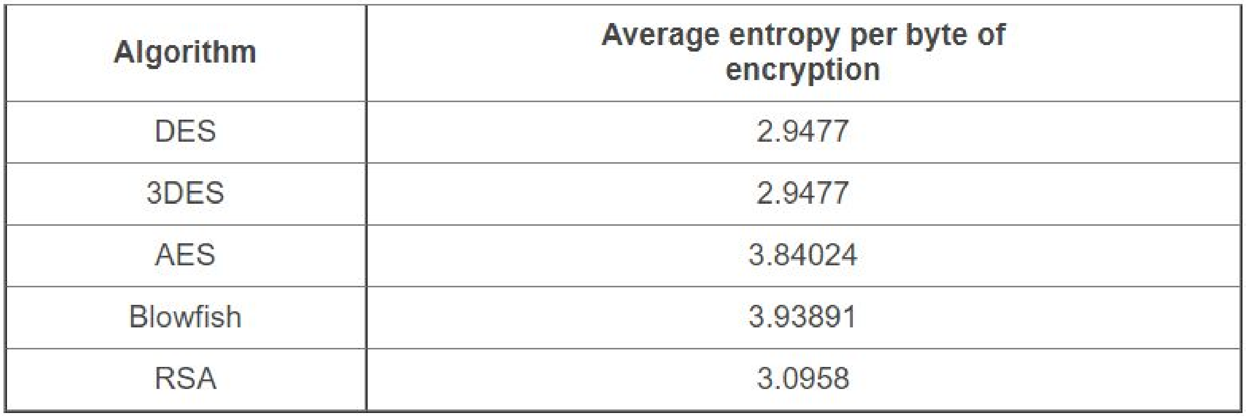


Algorithm	Memory used (KB)
DES	18.2
3DES	20.7
AES	14.7
Blowfish	9.38
RSA	31.5

Entropy Comparison (AES vs DES)

The entropy test checks for randomness & possibility of patterns. It’s measured using entropy per byte of encryption & is considered better when higher





Algorithm	Average entropy per byte of encryption
DES	2.9477
3DES	2.9477
AES	3.84024
Blowfish	3.93891
RSA	3.0958

We can notice that AES performs much better than DES when it comes to randomness.

Possible Attacks

While the AES has never been broken before, the famous whistleblower “Edward Snowden” revealed in 2013 that the NSA was extensively working on a method to crack several encryption algorithms, AES being a major one of them. Some of the unsuccessful attacks so far are:

a. XSL attacks tried to target the algorithm by considering it a system of quadratic equations; however, this attack did not even come close to breaking the cipher
b. Side-Channel attacks target some implementations of AES with low complexities starting from 2^{119} and then improved to $2^{99.5}$. While this might sound as a breakthrough, this attack might only work on simple or incorrect implementations of the algorithm while the correct implementation of AES remains unbreakable.
c. Another attack posted by Bruce Schneier targeting “AES-256” uses only two related keys and 2^{39} time to retrieve the full “256-bit” key of a “9-round version”, or 2^{45} time for a “10-round version” with a more powerful method of “related subkey attack”, or 2^{70} time for an “11-round version”. However, since “256-bit AES” uses “14 rounds”, these attacks are not still not effective against the full implementation of the AES algorithm.

Why AES Replaced DES

a. Security:
The impenetrable security of AES fulfilled the original aim of finding an algorithm that improves the security issue of DES.

b. Efficiency: The National Institute of Standards & Technology (NIST) has nominated AES as an algorithm with great computational efficiency allowing it to be utilized in a variety of applications especially in “high-speed broadband links”.

c. Implementation Characteristics:
Some of the characteristics of this algorithm relating to real-life implementation are flexibility, sustainability & simplicity whether implementation takes place in a software or hardware environment.

VI. Conclusion

Using the internet and network are growing rapidly. Every day, lots and lots of digital data is being exchanged between users. A considerable amount of the exchanged information includes secret or confidential data that needs to be protected. Encryption algorithms play vital roles to protect original data from unauthorized access and we do have more than a couple of existing ones. Advanced encryption standard (AES) algorithm is one of the most efficient algorithms and it is widely supported and adopted on hardware and software. As mentioned throughout the paper, what makes this method special is its ability to deal with different key sizes such as 128, 192, and 256 bits with 128 bits block cipher. Another noticeable thing regarding the AES algorithm is that the encryption and decryption processes are pretty similar except for a few variations and order difference. Several important features of the AES algorithm were discussed and the performance was evaluated based on previous researches and results. According to the results obtained from researches, AES can provide much more security compared to other algorithms like DES, 3DES, etc…. The AES algorithm got some weaknesses for sure, but they are minimal when compared to the strengths of it. Finally, we do not expect to see any change in the use of this algorithm in the near future unless quantum/supercomputers manage to break the cipher.

References

1. Bhargav, S., Majumdar, A., & Ramudit, S. (2008, Spring). 128-bit AES decryption. Retrieved November 21, 2020, from http://www.cs.columbia.edu/~sedwards/classes/2008/4840/reports/AES.pdf
2. Bruce Schneier (2009-07-30). "Another New AES Attack". Schneier on Security, A blog covering security and security technology. Archived from the original on 2009-10-05. Retrieved 2010-03-11.
3. Clark, A. (2018, August 2). How much encryption is too much: 128, 256 or 512-bit? Retrieved November 21, 2020, from https://discover.realvnc.com/blog/how-much-encryption-is-too-much-128-256-or-512-bit
4. E. Fernando, D. Agustin, M. Irsan, D. F. Murad, H. Rohayani and D. Sujana, "Performance Comparison of Symmetries Encryption Algorithm AES and DES With Raspberry Pi," 2019 International Conference on Sustainable Information Engineering and Technology (SIET), Lombok, Indonesia, 2019, pp. 353-357, doi: 10.1109/SIET48054.2019.8986122.
5. Kak, A. (2020, May 7). Lecture 8: AES: The Advanced Encryption Standard Lecture Notes on “Computer and Network Security”. Retrieved from https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
6. Mustafeez, A. Z. (n.d.). What is the AES algorithm? Retrieved November 20, 2020, from https://www.educative.io/edpresso/what-is-the-aes-algorithm
7. Nazeh Abdul Wahid MD, Ali A, Esparham B, Marwan MD (2018) A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention. J Comp Sci Appl Inform Technol. 3(2): 1-7. DOI: 10.15226/2474-9257/3/2/00132
8. SPIEGEL. (2014, December 28). Inside the NSA's War on Internet Security - DER SPIEGEL - International. Retrieved November 21, 2020, from https://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html
9. Thakkar, J. (2020, June 2). DES vs AES: Everything to Know About AES 256 and DES Encryption. Retrieved November 21, 2020, from https://sectigostore.com/blog/des-vs-aes-everything-to-know-about-aes-256-and-des-encryption/
10. Townsend Security. (2020, June 1). AES vs. DES Encryption: Why AES has replaced DES, 3DES and TDEA. Retrieved November 21, 2020, from https://www.precisely.com/blog/data-security/aes-vs-des-encryption-standard-3des-tdea
11. Wright, C. P., Dave, J., & Zadok, E. (2003, October). Cryptographic file systems performance: What you don’t know can hurt you. In Security in Storage Workshop, 2003. SISW'03. Proceedings of the Second IEEE International (pp. 47-47). IEEE.
12. Yenuguvanilanka, J., & Elkeelany, O. (2008, April). Performance evaluation of hardware models of Advanced Encryption Standard (AES) algorithm. In Southeastcon, 2008. IEEE (pp. 222-225).