Cryptography with

Advanced Encryption Standard (AES)

Jinil Chandarana – AU1940121 Keyur Nagar – AU1940207

Maulikkumar Bhalani – AU1940206 Umang Patel – AU1940174

Ahmedabad University

November 6, 2020

Abstract - The present report explained the implementation of AES cipher. Advanced Encryption Standard (AES) algorithm is one of the most sophisticated block ciphers used worldwide. The report explains crucial steps of encryption and decryption of plaintext and any other file format such as .txt, .pdf, .docx, .pptx, .xlsx, .mp4, .flv,.mp3, with example in each process along with some of the changes in the methods.

Key-words: AES, Rijndael, cryptography, block cipher, mix column, Galois field.

Introduction:

In this modern world, data plays an important role in our day-to-day life. securing data becomes a very hard job. For instance, Social media is a wide-spreading platform for engaging with sharing information such as pictures, private messages, mobile numbers, e-mail addresses, and so on. Net banking, data transfer, etc. rely on information and technology.[3] Wistfully, as these services use information, they can be manipulated. To overcome this issue, it becomes necessary to perform a certain operation on data using specific techniques so that it cannot be accessible except to the user.[2]

Cryptography is a method to secure information and communication technology derived from mathematical concepts and algorithms to transform by encrypting and then decrypting messages in ways that are hard to decipher. Some of such developed ciphers are Caesar cipher, monoalphabetic cipher, hill cipher, DES cipher, AES cipher, etc.

The cipher we use for cryptography is Advanced Encryption Standard (AES) in this project. Every mathematical operation taking place in this cipher is done in a finite field, Galois field GF (2^8). The data first converted into a 4x4 matrix which undergoes through the addition of round key and s box substitution (which does substitution), row shift and mix columns (which does permutation), and again the addition of the round key. As the key used in encryption opens up a minimum of 2^{128} possibilities. Owing to the reason, it is practically impossible for current computers to attack the cipher. Our project works with plaintext and any other file format encryption and decryption.[5]

LA concepts:

Following are the LA concepts used in the project. A detailed explanation of the concepts and how it works will be explained along with the approach

1. Galois field – each of the elements used in the algorithm are part of and performed under Galois field.
2. Non-linear transformation – substitution of each element from s-box describes the non-linear transformation of each element.

3. Linear transformation – linear transformation of each Column under Galois field through a MDS transformation matrix.
4. Addition under Galois field - for round key generation
5. Permutation – shifting row of matrix

Galois field

Computer stores data in bytes and hence, each binary number must be 8 bits long (1 byte = 8 bits). This is the reason why using GF(2^8) is best suitable for our algorithm. Each number in GF(2^8) the elements are of 8 bits comprising of 0's and 1's starting from 00000000 all the way to 11111111 hence, in total there are 256 elements in the field as the biggest byte is 1111111 = $2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255$. Since there are exactly 255 in ASCII each element can be uniquely assigned an ascii value.[1]

Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr
32	Space	48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	”	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	’	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

Fig: representation of ASCII

where Dec indicates the decimal representation (which can be converted to binary and stored as a byte) and Chr stands for character.

Each element is a byte and each element is represented as a polynomial.

For instance,

00000001 = 1

00000010 = x

00000011 = x + 1

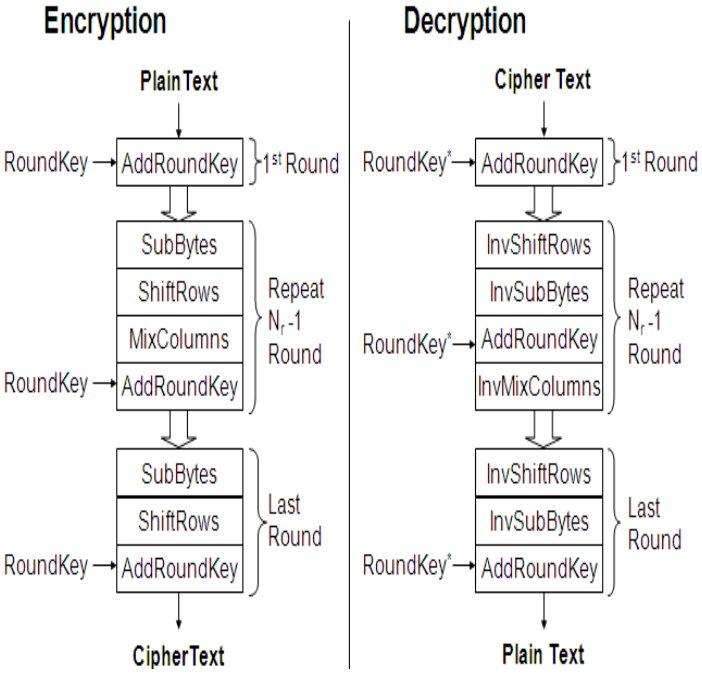
00000100 = x^2

00000101 = $x^2 + 1$

We perform addition, subtraction, multiplication and division (or inverse) to these elements. Whichever operations we perform, we stay in this field we never undergo or overflow the field. However, the mathematical operations are not the usual ones. Here, we will focus on addition corresponds to bitwise XOR (⊕), multiplication corresponding to bitwise AND (•).

Approach:

AES algorithm is a block cipher with symmetric key type which follows a particular structure for encryption and inverse process for decryption to provide the best security. There are three types of AES: AES-128, AES-192, AES-256, the difference lies in the number of rounds and in the length of the key. Each of the round comprises 4 sub processes namely, **substitution bytes, row shift, mix columns** and **add round key** all these processes are performed in finite field, Galois field GF(2^8). AES-128 consists of such 10 rounds, AES-192 consists of 12 rounds, AES-256 consists of 14 rounds. The data is first converted to a 128-bit block which is a 4x4 matrix. Each of these rounds encrypt and then decrypt a 128-bit block of data.



Following is the process of encryption and decryption of plaintext in AES-128.

Encryption:

Before going through the four processes of each round the plaintext is converted into a 4x4 matrix, now we will call it “State matrix”. Each element of the matrix is best represented in hexadecimal form of 8-bit or 1 byte. Similarly, the key (also a plain text) is converted into a 4x4 matrix. Further, each of the elements of the **state matrix** is XORed with the corresponding element of the **key matrix.**

For instance,

Plaintext: Two One Nine Two

T	w	o		O	n	e		N	i	n	e		T	w	o
54	77	6F	20	4F	6E	65	20	4E	69	6E	65	20	54	77	6F

Plaintext in Hex: 54 77 6F 20 4F 6E 65 20 4E 69 6E 65 20 54 77 6F

Forming matrix

$$\begin{pmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{pmatrix}$$

Key: Thats my Kung Fu

T	h	a	t	s		m	y		K	u	n	g		F	u
54	68	61	74	73	20	6D	79	20	4B	75	6E	67	20	46	75

Key in Hex: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Forming matrix

$$\begin{pmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{pmatrix}$$

.

Round Key expansion (Key Schedule):

As mentioned earlier, there are ten rounds in the AES-128 encryption. Each round has its own round key that is derived from the original AES-128-bit encryption which is described this section

AES algorithm is based on AES key expansion to encrypt and decrypt data. It is another most important step in AES structure. The key expansion routine creates round keys word by word, where a word is a vector of 4 bytes. The routine creates 4 x (N+1) words. Where N is the number of rounds. 4 words associated with each round.

Consider the given key matrix, In this matrix each column is represented by W[0], W[1], W[2], W[3] respectively.

W[0] = (54, 68, 61, 74)
W[1] = (73, 20 6D, 79)
W[2] = (20, 4B, 75, 6E)
W[3] = (67, 20 46, 75)

There are several steps to generate round keys which is described below:

1. circular byte Left shift:

In this step, each byte of the last column carried out a circular left shift. All other words remain the same. It will give us a new state matrix.

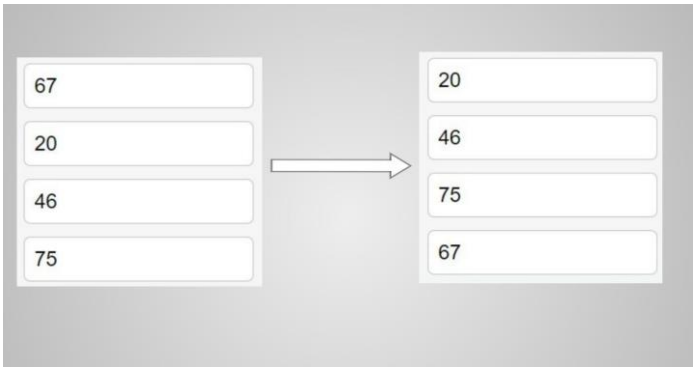


Fig: Circular left shift of third column

W[3] = (20,46,75,67)

2. Byte substitution:

Each of the element of third column(word) of state matrix will be substituted by the given Rijndael 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

Fig: Rijndael s-box

54	73	20	67
68	20	4B	20
61	6D	75	46
74	79	6E	75

For the above example, each element of the third column is substituted as follows.

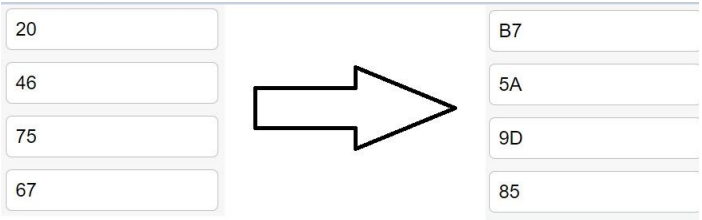


Fig: Substitution of last column

hence, the third column of state matrix, W[3] = (B7,5A,9D,85)

3. Adding round constant:

RCon specially generated vectors to perform the key expansion. Each of the Rcon is expressed by using indeterminant X^{i-1} . X^{i-1} is then modulo multiplied by prime, where prime = $x^4 + x + 1$. This generates an element under GF(2^8). This element is the first element of a round constant vector. Whereas all the other elements of the vector are zero.

For instance, for RC_3 we have indeterminate x^2,

$X^0 \bmod (x^4 + x + 1) = 1$ which is 00000001 under GF(2^8) and 01_{16} in hexadecimal.

Therefore vector RC_1 = [01,00,00,00]

RC_1	$\rightarrow x^{1-1}$	$=x^0$	mod *prime*	$=1$	$\rightarrow 00000001 \rightarrow 01_{16}$
RC_2	$\rightarrow x^{2-1}$	$=x^1$	mod *prime*	$=x$	$\rightarrow 00000010 \rightarrow 02_{16}$
RC_3	$\rightarrow x^{3-1}$	$=x^2$	mod *prime*	$=x^2$	$\rightarrow 00000100 \rightarrow 04_{16}$
RC_4	$\rightarrow x^{4-1}$	$=x^3$	mod *prime*	$=x^3$	$\rightarrow 00001000 \rightarrow 08_{16}$
RC_5	$\rightarrow x^{5-1}$	$=x^4$	mod *prime*	$=x^4$	$\rightarrow 00010000 \rightarrow 10_{16}$
RC_6	$\rightarrow x^{6-1}$	$=x^5$	mod *prime*	$=x^5$	$\rightarrow 00100000 \rightarrow 20_{16}$
RC_7	$\rightarrow x^{7-1}$	$=x^6$	mod *prime*	$=x^6$	$\rightarrow 01000000 \rightarrow 40_{16}$
RC_8	$\rightarrow x^{8-1}$	$=x^7$	mod *prime*	$=x^7$	$\rightarrow 10000000 \rightarrow 80_{16}$
RC_9	$\rightarrow x^{9-1}$	$=x^8$	mod *prime*	$=x^4+x^3+x+1$	$\rightarrow 00011011 \rightarrow 1B_{16}$
RC_{10}	$\rightarrow x^{10-1}$	$=x^9$	mod *prime*	$=x^5+x^4+x^2+x$	$\rightarrow 00110110 \rightarrow 36_{16}$

fig: formation of RCon

Fixed round constants corresponding to round numbers will be XORed by new W[3]. At Each round of the process, the round constant will be changed.

g(W[3]) = (B6,5A,9D,85)

For new round key,

W[4] = W[0] ⊕ g(W[3]) = (E2, 32, FC, F1)

0101 0100 1011 0110	0110 1000 0101 1010	0110 0001 1001 1101	0111 0100 1000 0101
1110 0010 E2	0011 0010 32	1111 1100 FC	1111 0001 F1

w[5] = w[4] ⊕ w[1]

w[6] = w[5] ⊕ w[2]

w[7] = w[6] ⊕ w[3]

therefore, w[4], w[5], w[6], w[7] will act as a new key 4x4 matrix.

w[n] = w[n-1] ⊕ w[n-4]

hence, we will get our first round key as follows:

$$\begin{pmatrix} E2 & 91 & B1 & D6 \\ 32 & 12 & 59 & 79 \\ FC & 91 & E4 & A2 \\ F1 & 88 & E6 & 93 \end{pmatrix}$$

Similarly, this process will be repeated till 10 times and generate a key schedule of 40 words (10 matrices).

(Round 0 is the original key)

- Round 0: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- Round 1: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
- Round 2: 56 08 20 07 C7 1A B1 8F 76 43 55 69 A0 3A F7 FA
- Round 3: D2 60 0D E7 15 7A BC 68 63 39 E9 01 C3 03 1E FB
- Round 4: A1 12 02 C9 B4 68 BE A1 D7 51 57 A0 14 52 49 5B
- Round 5: B1 29 3B 33 05 41 85 92 D2 10 D2 32 C6 42 9B 69
- Round 6: BD 3D C2 B7 B8 7C 47 15 6A 6C 95 27 AC 2E 0E 4E
- Round 7: CC 96 ED 16 74 EA AA 03 1E 86 3F 24 B2 A8 31 6A
- Round 8: 8E 51 EF 21 FA BB 45 22 E4 3D 7A 06 56 95 4B 6C
- Round 9: BF E2 BF 90 45 59 FA B2 A1 64 80 B4 F7 F1 CB D8
- Round 10: 28 FD DE F8 6D A4 24 4A CC C0 A4 FE 3B 31 6F 26

Substitution bytes:

The goal of the substitution step is to reduce the correlation between the input bits and the output bits at the byte level. The bit scrambling part of the substitution step ensures that the substitution cannot be described in the form of evaluating a simple mathematical function. This process performs non-linear transformation.

In this step, each of the elements represented in hexadecimal form is substituted with a fixed element from a lookup table, s-box. The first digit of hexadecimal is the row of s-box and the second digit is the column. The corresponding element from the s-box is substituted and hence, a new 4x4 state matrix is formed.

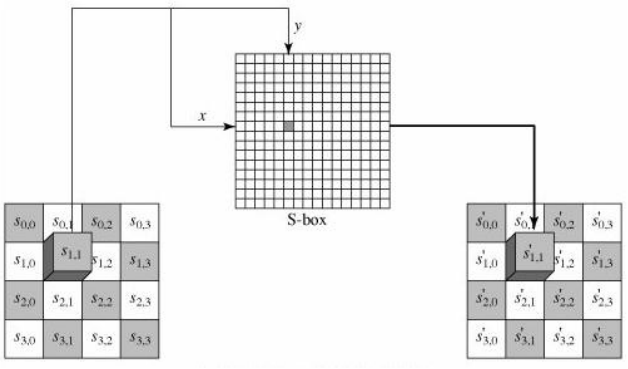


Fig: s-box substitution

For instance, byte 3C is substituted by entry of S-Box in row 3 and column C, i.e., by EB.

This leads to new state matrix,

$$\begin{pmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{pmatrix}$$

Row shift transformation

Row shift Is one of the essential parts of permutation in the cipher which diffuses the matrix. Under this process the bytes of the row undergo circular left shift excluding the zeroth row. The bytes of the row number zero remain the same, one circular left shift is carried out in first row, second row is shifted two bytes to the left and the third row is shifted three bytes to the left. The number of elements remains the same and of size 16 bytes.

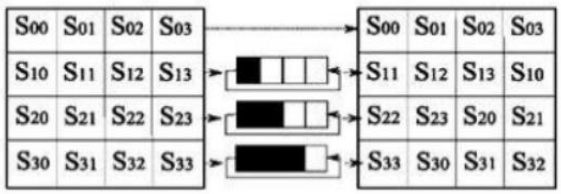


fig: row shift

for state matrix,

$$\begin{pmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{pmatrix} \rightarrow \begin{pmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{pmatrix}$$

Mix column transformation

MDS matrix

An MDS matrix (Maximum Distance Separable) is a transformation matrix over a finite field representing a linear transformation from G^n to G^m where G is a finite field. It has a certain diffusion property which helps mixing the bits of a particular element of a certain matrix, which is helpful in cryptography. [6]

Process

Along with the row shift, mix column transformation is a crucial step from diffusion of the state matrix. Each of the columns can be represented as a polynomial with coefficients in a finite field.

$$p(x) = p_3x^3 + p_2x^2 + p_1x + p_0$$

This will denote a word or vector in the form [p_3, p_2, p_1, p_0]. Here the polynomial is somewhat different from the polynomial used to define the elements of the Galois field, even though both of them use the same indeterminate, x. The coefficients used there are the elements of the Galois field themselves i.e. bytes, instead of bits.[6]

Architecture of MDS matrix

For the linear transformation of the state matrix, a predefined transformation matrix is used. The design criteria for the MDS matrix are

1. The matrix must have 4, 4 bytes columns hence, a 4x4 matrix.
2. The linear transformation is performed over GF(2^8).
3. It has to have a sufficient diffusion power.
4. The elements of the matrix have to be elements of GF(2^8).
5. The matrix has to be invertible.

The MSD used here is,

$$\begin{pmatrix} 02\,03\,01\,01 \\ 01\,02\,03\,01 \\ 01\,01\,02\,03 \\ 03\,01\,01\,02 \end{pmatrix}$$

Each of the vector of the state matrix multiplied by the transformation

$$GF(2^8) \rightarrow GF(2^8)$$

Let, each column of state matrix represents by four-term polynomial as

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

Let, each column of transformation matrix represents by four-term polynomial as

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

The transformation is carried out by multiplication of each of the vectors in the state matrix with each of the columns of the transformation matrix $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ modulo $x^4 + 1$. As multiplying two for term polynomial the resultant polynomial is a seven-term polynomial, which must be reduced to four-byte word, this is done by multiplication modulo $x^4 + 1$. $x^4 + 1$ is reducible polynomial over GF(2^8), therefore, multiplication by a fixed four-term polynomial is not necessarily invertible. However, the AES algorithm specifies a fixed four-term polynomial that does have an inverse. [4]

Thus, to transformation one vector is done by polynomial multiplication over Galois field

$$\begin{aligned} a(x) \bullet b(x) = c(x) &= \left(a_3x^3 + a_2x^2 + a_1x + a_0\right) \bullet \left(b_3x^3 + b_2x^2 + b_1x + b_0\right) \\ &= c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \end{aligned}$$

Where,

$$\begin{aligned} c_0 &= a_0 \bullet b_0 \\ c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 \\ c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \\ c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \\ c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\ c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\ c_6 &= a_3 \bullet b_3 \end{aligned}$$

To reduce the polynomial, we perform mod $\left(x^4 + 1\right)$

$$\begin{aligned} & c(x) \bmod \left(x^4 + 1\right) \\ &= \left(c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0\right) \\ &= c_6x^{6 \bmod 4} + c_5x^{5 \bmod 4} + c_4x^{4 \bmod 4} + c_3x^{3 \bmod 4} \\ &= c_6x^2 + c_5x + c_4 + c_3x^3 + c_2x^2 + c_1x + c_0 \\ &= c_3x^3 + \left(c_2 \oplus c_6\right)x^2 + \left(c_1 \oplus c_5\right)x + c_0 \oplus c_4 \\ &= d_3x^3 + d_2x^2 + d_1x + d_0 \end{aligned}$$

Where,

$$d_0 = c_0 \oplus c_4$$
$$d_1 = c_1 \oplus c_5$$
$$d_2 = c_2 \oplus c_6$$
$$d_3 = c_3$$

Thus, the transformed vector be represented as

$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$

the coefficient can be expressed as:

$$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$
$$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$$
$$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$$
$$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

Representing the coefficient of a(x):

$$d_0 = 2 \bullet b_0 \oplus 3 \bullet b_1 \oplus 1 \bullet b_2 \oplus 1 \bullet b_3$$
$$d_1 = 1 \bullet b_0 \oplus 2 \bullet b_1 \oplus 3 \bullet b_2 \oplus 1 \bullet b_3$$
$$d_2 = 1 \bullet b_0 \oplus 1 \bullet b_1 \oplus 2 \bullet b_2 \oplus 3 \bullet b_3$$
$$d_3 = 3 \bullet b_0 \oplus 1 \bullet b_1 \oplus 1 \bullet b_2 \oplus 2 \bullet b_3$$

therefore,

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Similarly, Performing the transformation for each vector in state matrix, we get a new state matrix

For instance,

$$\begin{pmatrix} 02\,03\,01\,01 \\ 01\,02\,03\,01 \\ 01\,01\,02\,03 \\ 03\,01\,01\,02 \end{pmatrix} \begin{pmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{pmatrix} = \begin{pmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{pmatrix}$$

Add Round Key:

The round key is added to the state matrix by a simple XOR operation. Each round key is consisting of 4 words from the key schedule as explained before. Each of the element of the state matrix is XORed with the corresponding element in the key matrix. The key matrix changes according to the round in for loop.

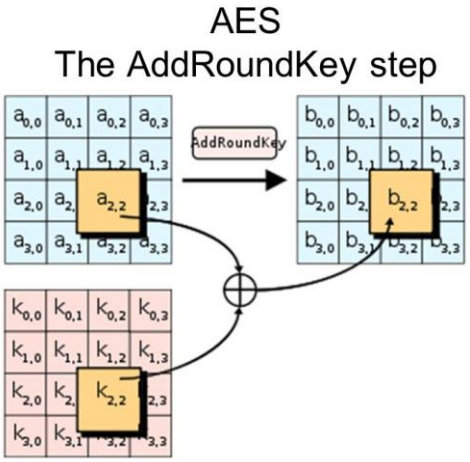


Fig: adding round key

addition of round key in state matrix is shown below

$$\begin{pmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{pmatrix} \rightarrow \begin{pmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{pmatrix}$$

fig: add round key

The above four processes (**substitution bytes, row shift, mix columns** and **add round key**) is carried out 9 times in a loop providing a great diffusion. The last round, round 10, is carried out without the MixColumns transformation to make it look similar going the reverse direction for decryption which makes it easier to understand during decryption.

Decryption:

The same above process is carried out with encrypted data with the help of inverse substitution box, right shift row, inverse mix column and add round key.

Other file format:

The project provide function for user to encrypt and decrypt file formats such as **.txt, .pdf, .docx, .pptx, .xlsx, .mp4, .flv .mp3** and many more. As these file formats have different formats the transformation for encryption and decryption becomes hazardous. As a result these files are converted into ASCII strings and the same process for encryption and decryption is carried out as of plaintext with the help of loop.

Coding and simulation

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])
- end for
- SubBytes(state)
- ShiftRows(state)
- AddRoundKey(state,&w[40])

Coding strategy

AES is a standardized cipher for protection of data worldwide. It was first introduced in 2003, since then many changes have been done to improve the approach and security of the cipher. We however, have represented one of the basic algorithms which is most commonly used. Most of the concepts used in all the main processes are derived from the algorithm obtained from several research papers.

strToArr():

Hexadecimal is represented in two digits for an 8-bit number, the first digit of hex represents the row of the substitution box and the second digit represents column of substitution box, it becomes easy to substitute elements hence conventionally hexadecimal representation is used. However, as generating a 16x16 lookup matrix having individual elements different, we generated an s box as a tuple and converted the plaintext (ASCII) into integer and formed a 4x4 matrix. As tuple consists of a specific index, each of the indexes can map to the s box element according to it's value from the plaintext matrix formed (state matrix).

```
def strToArr(s):
    array = [[0 for i in range(4)] for j in range(4)]
    i = 0
    ch = 0
    for i in range(4):
        for j in range(4):
            current = int(ord(s[ch]))

            array[j][i] = current
            ch = ch + 1
    return array
```

files to string

different file have different file formats. Hence, converting pixels to arrays according to the algorithm is quite hideous. To make it easy we used encoding file in **base64**. Base64 converts image into binary format and then into ASCII, making it easy to manage and hence, we are able to use the same algorithm as plaintext to encrypt and decrypt images using for loop to generate manouris matrices.

```
with open(IFile, "rb") as img_file:
    temptext = base64.b64encode(img_file.read())
temptext = temptext.decode('utf-8')
```

Fig: python code to convert inputFile to string

Key schedule matrix:

It is unsafe to store key matrices in the ROM because there are high chances to steal the data. Because once the data is encrypted, the file which consists of all-round keys will always be in the system. So, it is necessary to store data in the RAM rather than store it in ROM. So, we have used a 4x44 key matrix to store all round keys. Each time of the execution of encryption and decryption of the program, all round keys will be generated and stored in the key schedule matrix.

Login():

The first execution of our program, it will take username and password. The important point to note is that our password works like a key. It was very important to do because if every time a user has to give a key and encryption will be done according to this key then whenever the user has to decrypt data, he/she has to give the right key. Otherwise, it will do wrong decryption. For that reason, we have taken one password then it will continue for all time. If a user gives the wrong password or username, then he/she cannot do encryption or decryption of data. so, it will be applicable for one user only.

invMixColumn()

Inverse of transformation matrix is essential in decryption in order to obtain the transformation back from co-domain to domain. In order to do so the inverse of the transformation matrix is essential. However, failing to learn the mathematical process to obtain inverse in Galois field and lack of information about the same, we were able to perform Inverse function and find the inverse of the matrix. Instead we have simply defined the inverse matrix and performed the operation.

flow

As the secrecy of the data is the main concern the user has to login before proceeding, here the password works as a key from the algorithm. After login the user will select function **encrypt** or **decrypt.** Further according to the previous choice users have to opt for plaintext or image. The input will be converted into a matrix followed by the four processes – **S-box substitution (non-linear transformation), row shift (permutation), mix Column (linear transformation)** and **add round key (addition under GF(2^8).** the encrypted output data in the form of unrecognizable ASCII characters is stored as an .exe file. The same process works in the reverse order for decryption of the data.

Individual contribution:

Keyur Nagar:

file I/o of text , all of the key schedule(round keys) generated and stored in the proper way by him. Image decryption method

Jinil Chandarana

mix column method and file to string conversion or string to file conversion done by him. Image encryption method.

Maulikkumar Bhalani:

Login() method, row shifting method and file I/o of login done by him.

Text encryption method

inverse mix column

Umang Patel:

Add round constant, substitution method and system argument part done by him.

Text decryption method

inverse substitution

References

1. Benvenuto, C. J. (2012, may 31). *Galois Field in Cryptography*. Retrieved from https://sites.math.washington.edu/: https://sites.math.washington.edu/~morrow/336_12/papers/juan.pdf
2. medium. (2019, november 08). *Real Life Applications*. Retrieved from medium.com: https://medium.com/@prashanthreddyt1234/real-life-applications-of-cryptography-162ddf2e917d
3. N/A. (2001, november 26). *Specification for the AES*. Retrieved from https://csrc.nist.gov/: https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf
4. springer. (2020, july 05). *MixColumns Coefficient Property*. Retrieved from springer.org: https://link.springer.com/chapter/10.1007/978-3-030-51938-4_6

5. wikipedia. (2009, october 23). *Advanced Encryption Standard*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
6. wikipedia. (2020, April 07). *MDS_Matrix*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/MDS_matrix