Enhanced AES algorithm implementation using different key size

This assignment for the SP.TP.Information security course was done by the following the students :

J. Houssem ; E. Bora ; D. Yunus Emre

Abstract

- As the technology advances the amount of data we exchange increases tremendously as well. In order to ensure the safety of the data we use certain cryptology algorithms that take the data and convert it to a text that is almost impossible to understand by intruders. AES is used to encrypt or decrypt a 128-bit input. AES stands for Advanced Encryption System and it is a symmetric encryption algorithm. It’s predecessor DES was being used before it however it was vulnerable to brute force attacks and it’s key size was 56 bits which was not long enough. Hence AES was created as an algorithm which is robust against brute force attacks. The AES algorithm uses a 128-bit symmetric block cipher that encrypts and decrypts a given data. The AES encryption process creates a ciphertext which is a text that humans couldn’t interprete. It is an indecipherable conversion from the plaintext data. The output of the encryption process, the AES ciphertext, cannot be read until the same secret key is used to decrypt it. That is why it is called symmetric. At encryption and decryption 128-bit, 192-bit, and 256-bit keys can be used to convert a plaintext into a ciphertext or a ciphertext into a plaintext. AES does its operations on bytes rather than bits that is why it is more efficient compared to DES. AES treats the 128 bits of a given plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows known as “the state” to obtain a matrix on which several operations would be performed.

I. Introduction

- In this technical world data are commuted through electronic media where data security and privacy are the major problem to be considered. In many cases multimedia applications require to satisfy many aspects of transmission. Reliable security of data is needed to protect data from unauthorized user (CIA triad). To ensure security, cryptography is the only way for secure data transmission. One such algorithm of cryptography is Advanced Encryption Standard algorithm commonly known as AES algorithm, a symmetric algorithm using the same key (private key) for encryption and decryption. AES algorithm is known for its resistance against the known attacks, its design simplicity, fast data processing and code compactness. Netherless, it’s main weakness is, whoever has the key can read the message easily. The implemented algorithm for this assignment uses a fixed plaintext size (32 hex characters per line) and 128, 192 and 256 bits of key size. As the size of the key increases it is harder to decrypt the encrypted the plaintext. Therefore we also covered the 256 bits key size which ensures greater security. The number of rounds for processing the plain text is increased from 10 rounds (AES-128) to 14 rounds (AES-256), where the first 13 rounds perform all the four operations of the AES algorithm (will be explained in details) and the final round implements three of the four operations for encryption (without mixing the columns). Decryption is done by using the inverse operations of the encryption algorithm. AES will be suitable for applications with high security and throughput requirements thanks to its robust security and resilience against hacking attempts (the higher-size of the key, the more secure the AES could get). Moreover, AES has two different modes to encrypt and decrypt a text : Cipher Blocking Chaining (CBC) and Electronic CodeBook (EBC). CBC is an advanced form of block cipher enc/decyrption which adds an extra level of complexity to the encrypted data thanks to using an extra vector (not extra key) to encode the plaintext (more to it in the next slides)

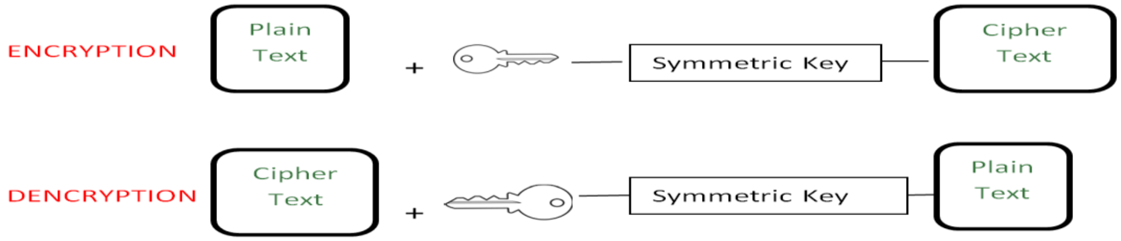


Fig 1. Cryptography

Cipher Text- Encrypted text in coded human unreadable form is called a cipher data which is the result of encryption performed on the original data. (exp : ‘Cryptography is fun’)

Encryption- It is the process of encoding information in such a way that only authorized parties can read it by denying the interception by others.

Decryption- It is the backward process of encryption by converting encrypted data back to its original form. It is generally the reverse process of encryption where only an authorized user can only decrypt the text as it requires a secret key.

Symmetric Cryptography (AES) It uses the same secret (private) key to encrypt and decrypt the data. It requires the secret key be known by the party encrypting the data and the party decrypting the data. The keys may be identical or there may be a simple transformation to go between the two keys.

II Algorithm Architecture

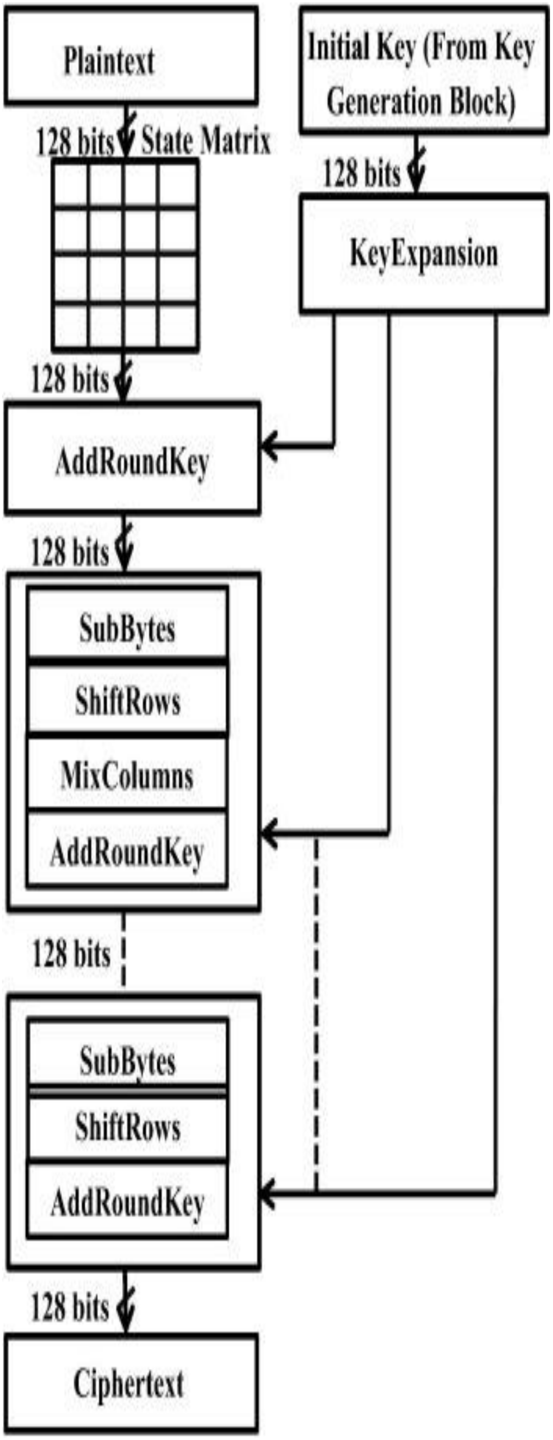


Fig 2. AES- 128 bits System Architecture

AES is a symmetric key cryptography and it is an iterated block cipher with a fixed block size of 128 bits and a variable key length i.e. it may be 128, 192 or 256 bits. The different transformations operate on the intermediate results, called **state (see code).**

The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array of dimension 4x4 is used to store these 16 byte (hex) values.

AES uses a variable number of rounds, Nr which are fixed: A key of sizes 128, 192 and 256 bits have 10, 12 and 14 rounds respectively.

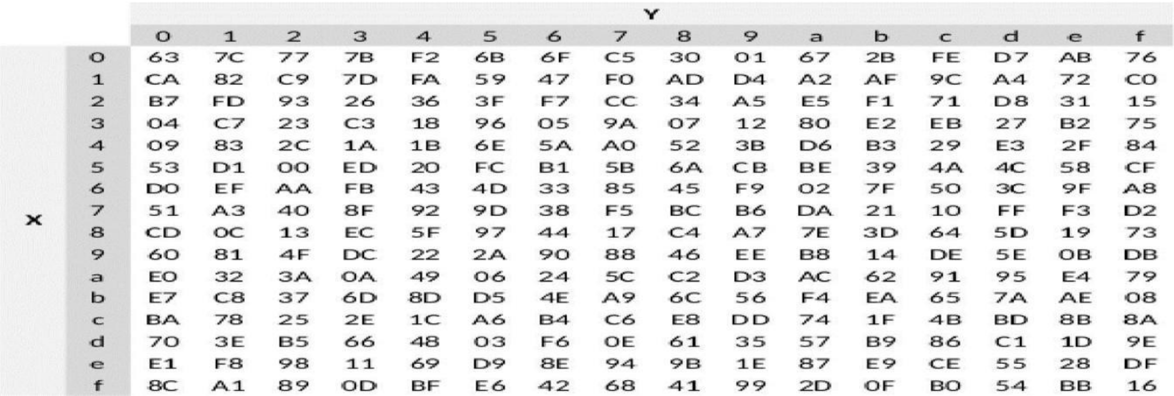
Since it is an asymmetric key cryptography, the Key(private key) which is got as input from the user is used for both encryption and decryption. Key undergoes an expansion process to involve in the encryption and decryption processes.

Each round has four operations (S-Box substitution, Shift Rows, Mix Columns and Add Round Key) except the final round (S- Box substitution, Shift Rows and Add Round Key) in both encryption and decryption process which is depicted in the following Figure 4.

III Implementation

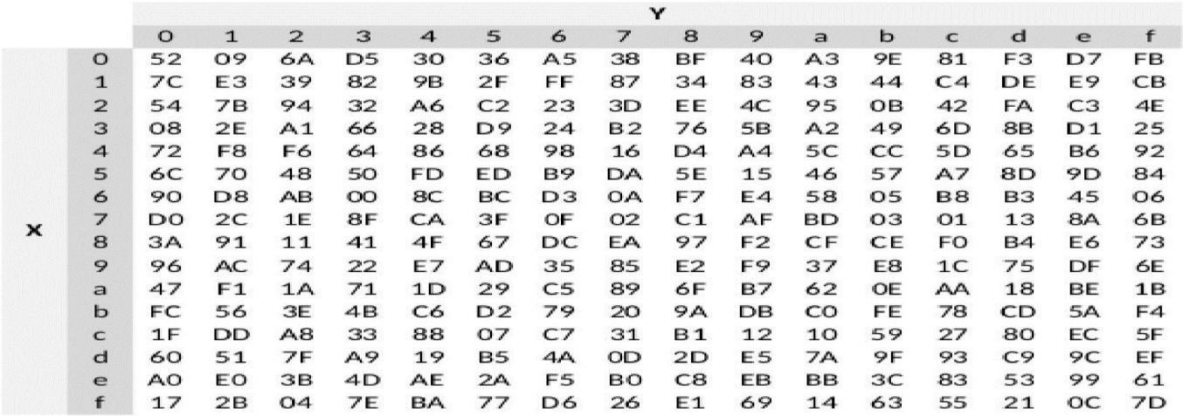
1. S-Box Substitution Module

The taken data that is read from the file as 32 hexadecimal numbers are converted to 16 bytes and those bytes are inserted in a 4x4 column-major order matrix. After the matrix is created, it’s elements are mapped into another matrix by using the Sbox as the transform matrix. Each byte has 2 hexadecimal numbers. The first one corresponds to the row of the Sbox matrix and the second one is the column number of the Sbox matrix and the intersection is our transformed byte information. During encryption each value of the state is replaced with the corresponding S-Box value. Many different block ciphers use this special substitution. S-Boxes provide an invertible transformation of segments of plaintext during encryption, with the reverse during decryption. (exp sub(19) = D4 : with x=1 and y=9)



		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

Fig 3.a Sub-Box Rijindael AES-based (encryption)

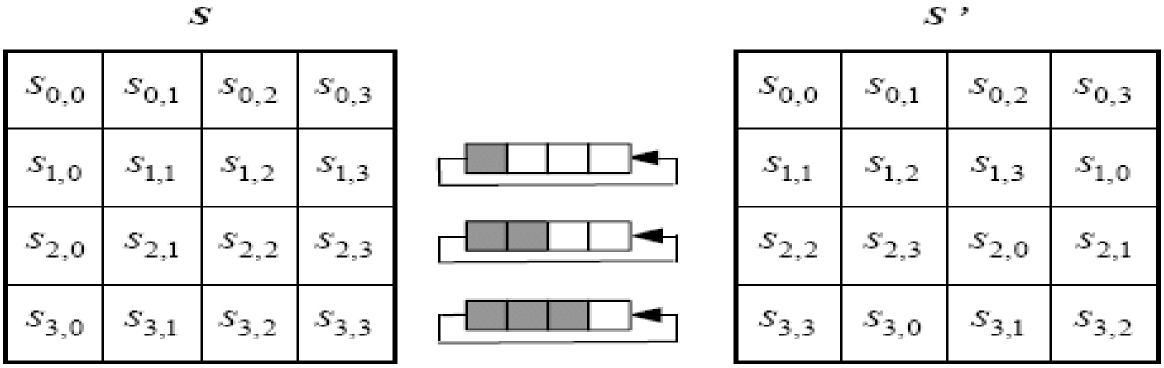


		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	83	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	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

Fig 3.b Inverse Sub-Box Rijindael AES-based (decryption)

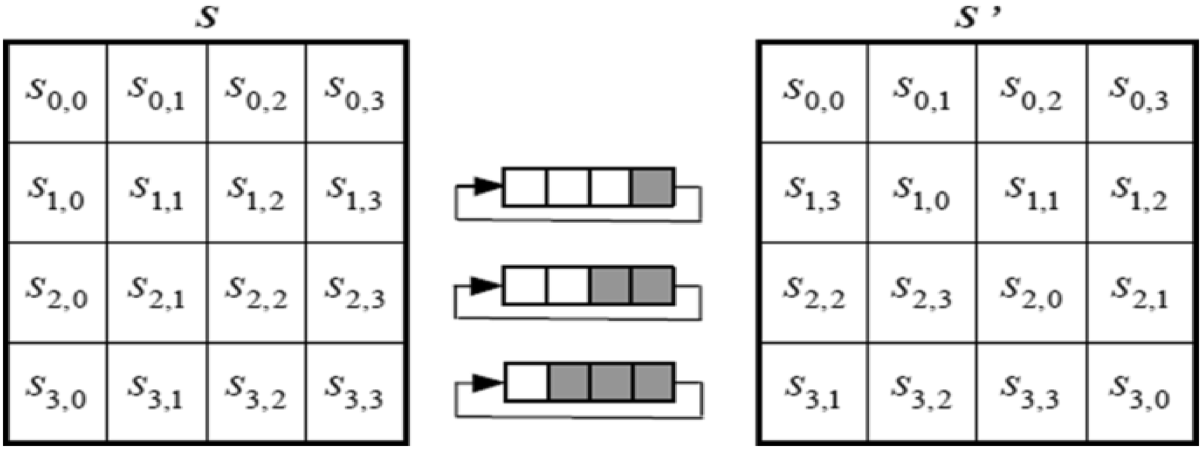
2. Shift Rows Module

After SubBytes, the newly obtained matrix’s some elements are shifted according to a certain pattern. The first row is left unchanged, the second one is shifted to the left by one element, the third one is shifted to the left by two elements and the fourth one is shifted to the left by three elements. When the elements on the first column are shifted left, they are inserted back at the last column of the same row. So it’s not a bit wise shift. The circular shift just moves each byte one space over. The matrix is formed vertically but shifted horizontally. The number of shifts made is with respect to the row number, where zeroth row has zero shift and first row with one shift, second row with two shifts and so on. (fig 4)



$$S = \begin{array}{|c|c|c|c|}\hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \qquad S' = \begin{array}{|c|c|c|c|}\hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ \hline s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ \hline s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \\ \hline \end{array}$$

Fig 4.a Shift Rows Operation



$$S = \begin{array}{|c|c|c|c|}\hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \qquad S' = \begin{array}{|c|c|c|c|}\hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,3} & s_{1,0} & s_{1,1} & s_{1,2} \\ \hline s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ \hline s_{3,1} & s_{3,2} & s_{3,3} & s_{3,0} \\ \hline \end{array}$$

Fig 4.b Inverse Shift Rows Operation

3. Mix Columns Module

In this step, each column of the matrix obtained from the ShiftRows is multiplied with a predefined matrix (second matrix in the fig 5) and this multiplication becomes the columns of the new matrix.

So The Mix Column transformation is multipling the columns of the data matrix by a pre-defined matrix.

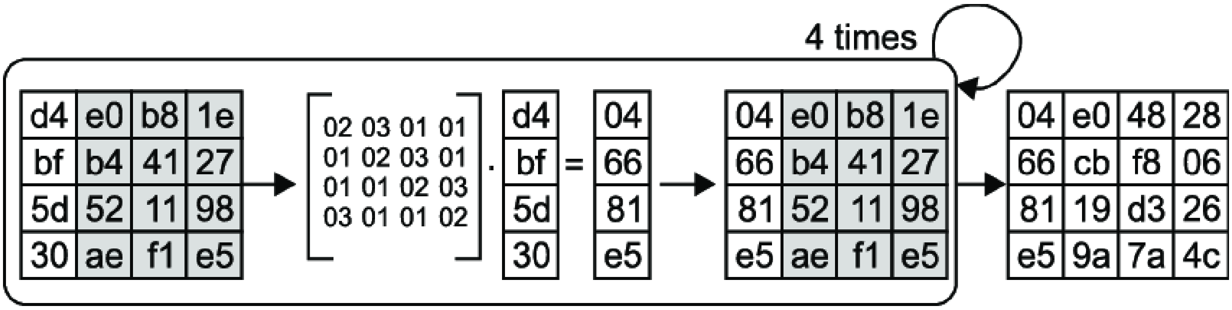
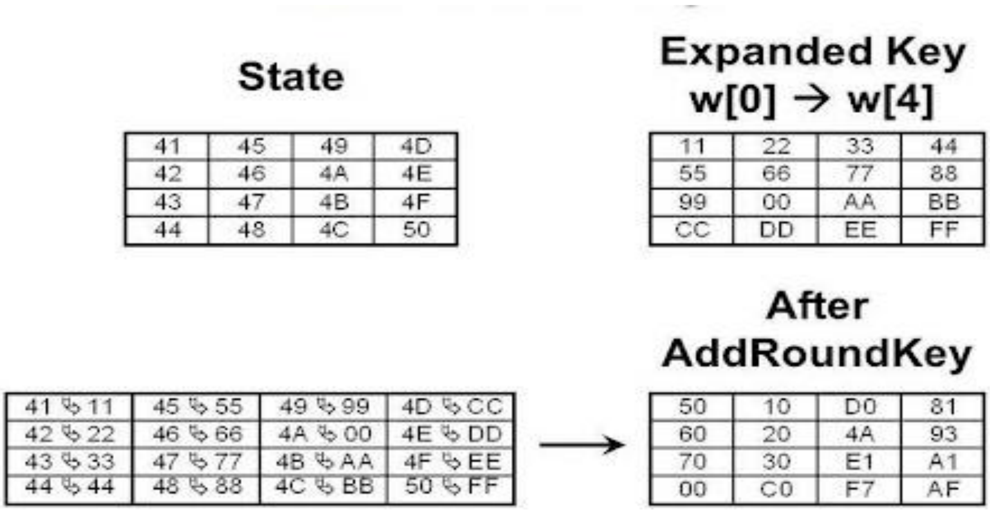


Fig 5. Mix Column

4. Add Round Key Module

In the AddRoundKey, the subkey is combined with the matrix created in the MixColumns parts. For each round, a subkey is derived from the main key using Rijndael's key schedule. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.
More detailed, Each time the Add Round Key function is called a different part of the expanded key is XORed against the state. In order for this to work the Expanded Key must be large enough so that it can provide key material for every time the Add Round Key function is executed. The Add Round Key function gets called for each round as well as one extra time at the beginning of the algorithm. Therefore the size of the expanded key will always be equal to
64* (number of rounds + 1)

The value 16 in the above function is actually the size of the block in bytes. This provides key material for every byte in the block during every round +1. The expanded key is of 64 * 11 = 704 bytes in size.



State

41	45	49	4D
42	46	4A	4E
43	47	4B	4F
44	48	4C	50

Expanded Key
w[0] → w[4]

11	22	33	44
55	66	77	88
99	00	AA	BB
CC	DD	EE	FF

41 ⊕ 11	45 ⊕ 55	49 ⊕ 99	4D ⊕ CC
42 ⊕ 22	46 ⊕ 66	4A ⊕ 00	4E ⊕ DD
43 ⊕ 33	47 ⊕ 77	4B ⊕ AA	4F ⊕ EE
44 ⊕ 44	48 ⊕ 88	4C ⊕ BB	50 ⊕ FF

After
AddRoundKey

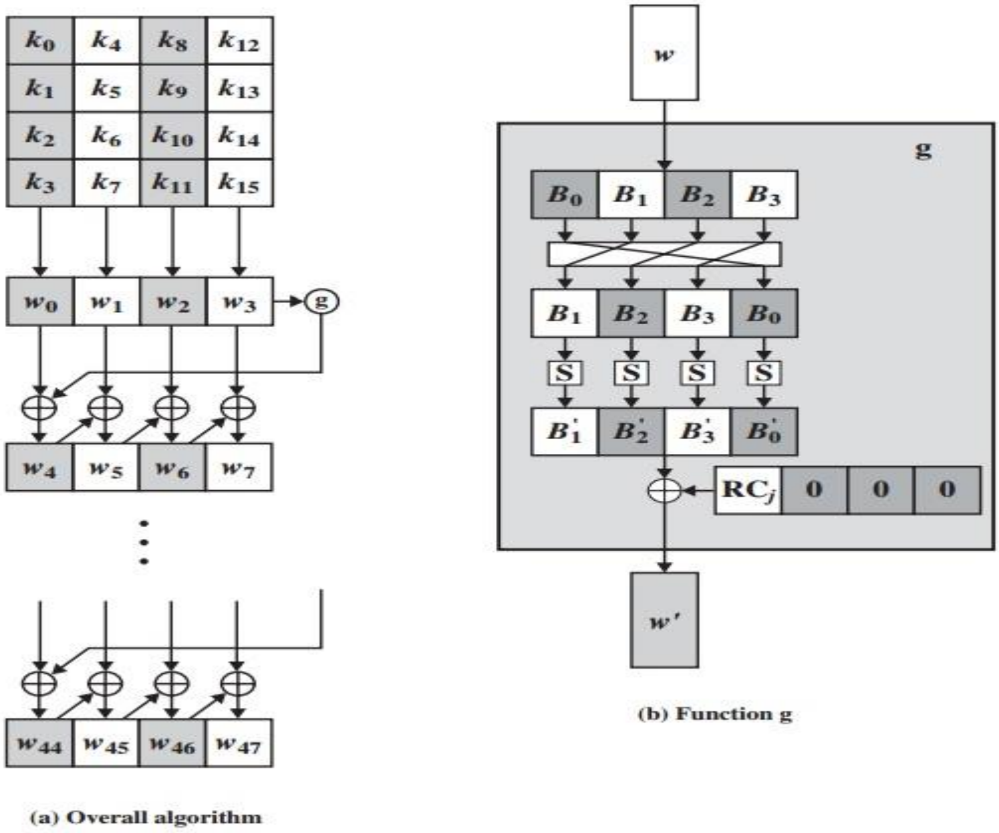
50	10	D0	81
60	20	4A	93
70	30	E1	A1
00	C0	F7	AF

Fig 6. Add round key

5. Key expansion

This step should be done at the beginning of the AES algorithm.

The AES algorithm takes the Cipher Key and performs a Key Expansion calculation to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words where Nb is the block size and Nr is the round number. The algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words.
During the step the algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The key is copied into the first four words of the expanded key.



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{44} w_{45} w_{46} w_{47}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B_1' B_2' B_3' B_0'
RC_j 0 0 0
w'

(a) Overall algorithm

(b) Function g

Fig 7. Key expansion

V AES modes ECB and CBC :

1 . The ECB (Electronic Code Book)

The ECB (Electronic Code Book) mode is the simplest of all. Due to obvious weaknesses, it is generally not recommended.

Every block will be encrypted with the same key and same algorithm. So if we encrypt the same plaintext, we will get the same ciphertext. So there is a high risk in this mode. And the plaintext and ciphertext blocks are a one-to-one correspondence. Because the encryption/ decryption is independent, so we can encrypt/decrypt the data in parallel. And if a block of plaintext or ciphertext is broken, it won’t affect other blocks.

In the database encryption, we can use ECB to encrypt the tables, indexes, wal, temp files, and system catalogs. But with the issues of security, this mode is not recommended.

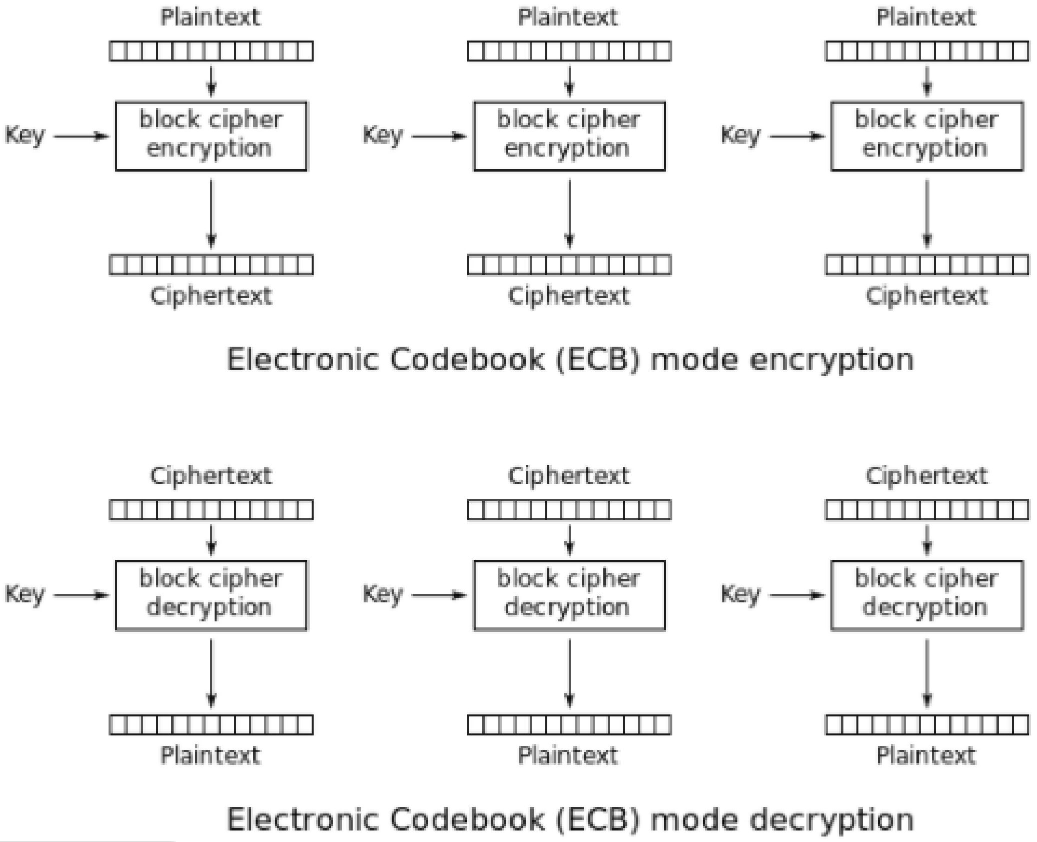


Fig 8. ECB mode

2 . The CBC (Cipher Block Chaining)

Unlike ECB this mode is using an initialization vector – IV. The IV has the same size as the block that is encrypted. In general, the IV usually is a random number. So it's vector not a second key (because AES is symmetric). Therefore this vector should generated randomly. Encyption and decryption on this mode should use the same vector IV.

We can see it in figure 2, the plaintext is divided into blocks and needs to add padding data. First, we will use the plaintext block xor with the IV.

Then CBC will encrypt the result to the ciphertext block. In the next block, we will use the encryption result to xor with plaintext block until the last block. In this mode, even if we encrypt the same plaintext block, we will get a different ciphertext block.

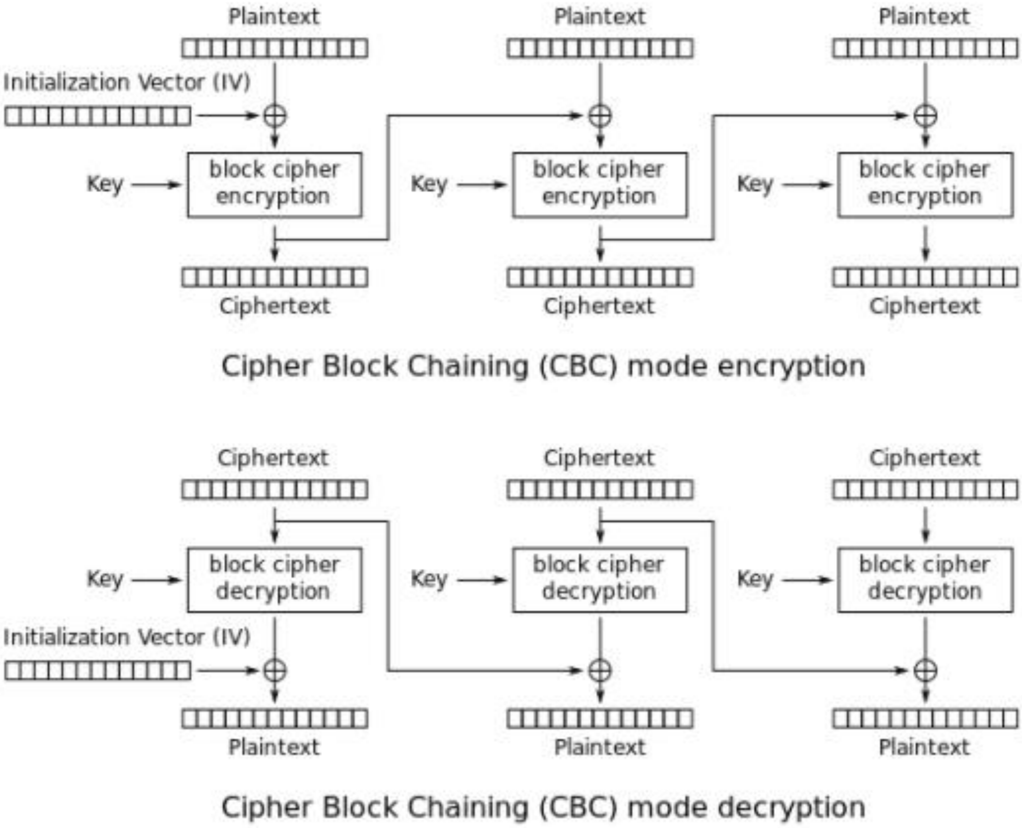


Fig 9. CBC mode

VI Experiment :

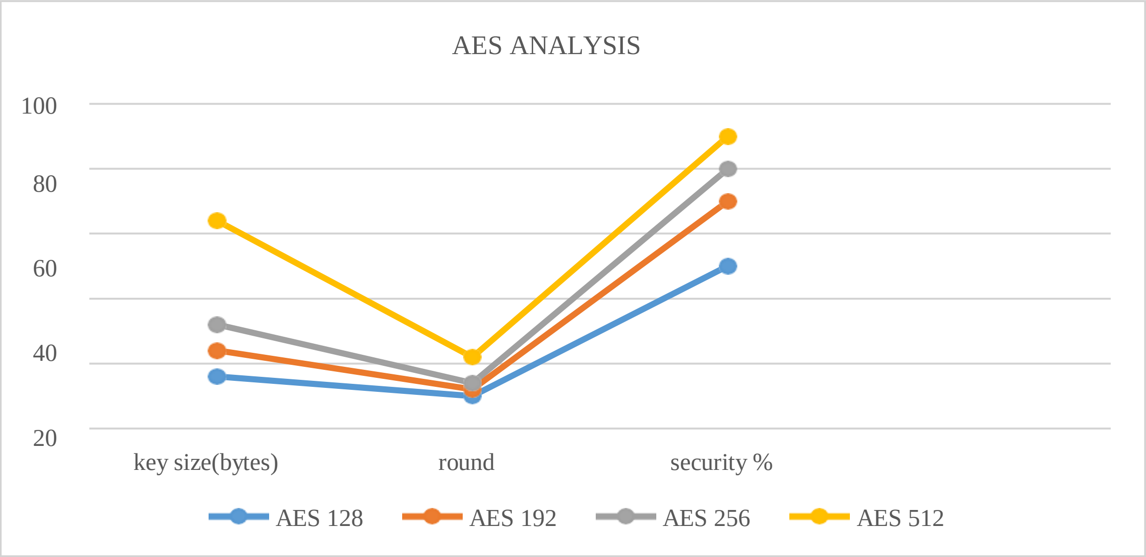


Fig 8. Output Performance

The above chart shows the analysis of the AES algorithm. Since the numbers of rounds are increased with increase in key size the security level gets increased than the existing algorithms. Besides using other different modes such as CBC, CFB, OFB increases the complexity for the operation and therefore the overall security.

IV Conclusion

In this project, AES is used to encrypt or decrypt a given input. The code can do the encryption and the decryption on a 128-bit input (32 hex characters) with keys that might have 128-, 192- and 256-bits using ECB or CBC with the format given in the instructions.

References

The followings websites, book, and youtube channels helped are deliver this assignment.

Youtube channles :

Advanced Encryption Standard [AES] - Kurz erklärt! (in german)

Bill Buchanan OBE : Encryption: ECB v CBC

AES Explained (Advanced Encryption Standard) - Computerphile

128 Bit or 256 Bit Encryption? - Computerphile

AES Rijndael Cipher explained as a Flash animation

Key Expansion in AES | Round Constant and g function in Key Expansion in AES

Animation of RIJNDEAL CIPHER : AES Encryption algorithm

Websites :

AES algorithm

AES Key Expansion

Advanced_Encryption_Standard (provided by the professor)

Bits, Bytes, Hexadecimals, and ASCII

stackoverflow

geeksforgeeks