|  |  |
| --- | --- |
| **Academic Year: 2024-25** | **Programme: BTECH-Cyber (CSE)** |
| **Year: 2nd** | **Semester: IV** |
| **Student Name: Jal Bafana** | **Batch : k1** |
| **Roll No: K005** | **Date of experiment: 27.02.2025** |
| **Faculty: Rejo Mathew** | **Signature with Date:** |

**Experiment 8: AES algorithm**

**Aim:** Write a program to implement AES algorithm.

**Learning Outcomes:**

After completion of this experiment, student should be able to

1. Describe working of AES algorithm.
2. Explain various modes of AES algorithm
3. Understand application of AES along with its advantage and limitations.

**Theory:**

Advanced Encryption Standard (AES) is a symmetric block cipher specification for the encryption of electronic data approved by the U.S National Institute of Standards and Technology (NIST) in 2001. AES operates on a 128-bit block size. The key size can be 128/192/256 bits. That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

|  |  |  |  |
| --- | --- | --- | --- |
| Key size (words/bytes/bits) | 4/16/128 | 6/24/192 | 8/32/256 |
| Number of rounds | 10 | 12 | 14 |
| Expanded key size (words/byte) | 44/176 | 52/208 | 60/240 |

The broad level steps are given below.

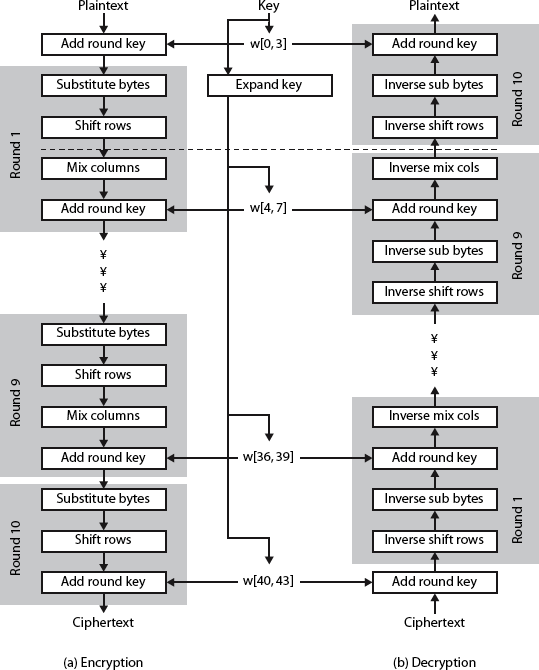
Do the following one time initialization processes:

* + Expand the 16 byte key to get the actual key block to be used
  + Do one time initialization of the 16 byte plain text block (called state)
  + XOR the state with the key block

For each round except final round, do the following:

* + Apply S-box to each of the plain text bytes (SubBytes)
  + Rotate row k of the plain text block (i.e state) by k bytes (ShiftRows).
  + Perform a mix column operation (MixColumns).
  + XOR the state with the key block (AddRoundKey)

(Note: in final round MixColumn operation is not performed)



**Procedure:**

1. Write a program to implement AES algorithm.
2. Refer to the class PPT and perform one round of AES Operation.
3. You can take the same input matrix and follow the steps given.
4. Create a word document for your observation and answer the following questions. Upload your document on MS Teams along with your code.

**Code: *type or copy your completed working code here***

from tabulate import tabulate

import numpy as np

# Function to display the matrix in a readable format

def print\_matrix(matrix):

formatted\_matrix = []

for row in matrix:

formatted\_row = [hex(value).replace('0x', '').zfill(2).upper() for value in row]

formatted\_matrix.append(formatted\_row)

print(tabulate(formatted\_matrix, tablefmt='fancy\_grid', stralign='center'))

print("")

# Replaces each value of the matrix with the corresponding S-Box value

def apply\_sbox(matrix):

sbox = [

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67,

0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59,

0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7,

0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1,

0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05,

0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, 0x09, 0x83,

0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,

0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b,

0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa,

0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,

0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc,

0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec,

0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,

0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee,

0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49,

0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,

0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4,

0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6,

0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, 0x70,

0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9,

0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e,

0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1,

0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,

0x54, 0xbb, 0x16

]

replaced\_matrix = [[0] \* 4 for \_ in range(4)]

for i in range(4):

for j in range(4):

replaced\_matrix[i][j] = sbox[matrix[i][j]]

return replaced\_matrix

# Function to shift the rows of the matrix

def shift\_rows(matrix):

matrix[1] = matrix[1][1:] + matrix[1][:1]

matrix[2] = matrix[2][2:] + matrix[2][:2]

matrix[3] = matrix[3][3:] + matrix[3][:3]

return matrix

# Galois Field multiplication used in MixColumns

def gf\_multiply(a, b):

a\_bin = list(map(int, bin(a)[2:].zfill(8)))

b\_bin = list(map(int, bin(b)[2:].zfill(8)))

result = np.polynomial.polynomial.polymul(a\_bin, b\_bin)

result = [int(x % 2) for x in result]

while len(result) > 8:

if result[-1] == 1:

result[0] ^= 1

result[1] ^= 1

result[3] ^= 1

result[4] ^= 1

result.pop()

else:

result.pop()

result = int(''.join(map(str, result[::-1])), 2)

return result

# MixColumns step of AES

def mix\_columns(matrix):

mix\_matrix = [

[2, 3, 1, 1],

[1, 2, 3, 1],

[1, 1, 2, 3],

[3, 1, 1, 2]

]

mixed\_matrix = [[0] \* 4 for \_ in range(4)]

for i in range(4):

for j in range(4):

for k in range(4):

mixed\_matrix[i][j] ^= gf\_multiply(matrix[k][j], mix\_matrix[i][k])

return [[value % 256 for value in row] for row in mixed\_matrix]

# AddRoundKey step of AES

def add\_round\_key(matrix):

round\_key = [

[0xac, 0x19, 0x28, 0x57],

[0x77, 0xfa, 0xd1, 0x5c],

[0x66, 0xdc, 0x29, 0x00],

[0xf3, 0x21, 0x41, 0x6a]

]

return [[matrix[i][j] ^ round\_key[i][j] for j in range(4)] for i in range(4)]

# Full AES encryption process

def aes\_encrypt(matrix):

print("Initial Matrix: ")

print\_matrix(matrix)

matrix = apply\_sbox(matrix)

print("After S-Box: ")

print\_matrix(matrix)

matrix = shift\_rows(matrix)

print("After Row Shifts: ")

print\_matrix(matrix)

matrix = mix\_columns(matrix)

print("After Mix Columns: ")

print\_matrix(matrix)

matrix = add\_round\_key(matrix)

print("After Adding Round Key: ")

print\_matrix(matrix)

for \_ in range(8):

matrix = add\_round\_key(mix\_columns(shift\_rows(apply\_sbox(matrix))))

print("After 8 Rounds: ")

print\_matrix(matrix)

matrix = add\_round\_key(shift\_rows(apply\_sbox(matrix)))

print("Final Round: ")

print\_matrix(matrix)

return matrix

# Sample input

input\_matrix = [

[0xea, 0x04, 0x65, 0x85],

[0x83, 0x45, 0x5d, 0x96],

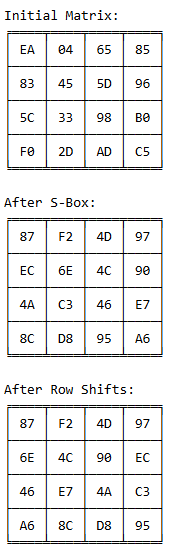
[0x5c, 0x33, 0x98, 0xb0],

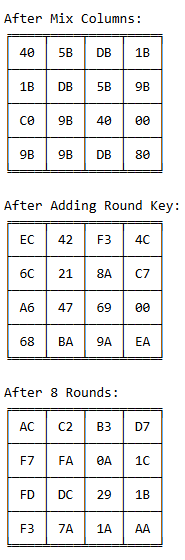
[0xf0, 0x2d, 0xad, 0xc5]

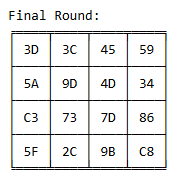
]

# Start encryption

output\_matrix = aes\_encrypt(input\_matrix)







**Questions:**

1. Describe working of AES algorithm in brief.

AES (Advanced Encryption Standard) is a symmetric encryption algorithm that operates on blocks of data. It uses a fixed block size of 128 bits and key sizes of 128, 192, or 256 bits. AES encryption involves several steps: key expansion, initial round (add round key), multiple rounds of substitution (using an S-box), shifting rows, mixing columns, and adding round keys. The number of rounds depends on the key size—10 rounds for 128-bit, 12 rounds for 192-bit, and 14 rounds for 256-bit keys. The encryption process ensures the confidentiality of data through these complex transformations.

1. Explain modes of operation of AES algorithm.

AES can operate in different modes to handle data of various sizes and to provide additional security features. Some common modes are:

* **ECB (Electronic Codebook):** Each block is encrypted independently, which makes it vulnerable to pattern recognition.
* **CBC (Cipher Block Chaining):** Each block is XORed with the previous ciphertext block before being encrypted, making it more secure than ECB.
* **CFB (Cipher Feedback):** Encrypts smaller increments of data, typically 8, 16, or 32 bits.
* **OFB (Output Feedback):** Similar to CFB, but the output of the encryption process is used as feedback for further encryption.
* **CTR (Counter):** Encrypts a counter value and XORs it with the plaintext, allowing parallel encryption and decryption.

1. Why does AES use the **Rcon (Round Constant) values** in key expansion?

AES uses the Rcon values during key expansion to provide a unique and non-repeating set of constants for each round. These values help in deriving the round keys and prevent key-related vulnerabilities. Rcon ensures that the keys in different rounds are diverse and significantly alters the transformation at each step, contributing to AES’s resistance to cryptographic attacks.

1. What makes **AES-256** more resistant to quantum computing attacks but vulnerable to side channel attacks

AES-256 is more resistant to quantum computing attacks compared to AES-128 because it has a larger key size, making it harder to break using Grover’s algorithm, which reduces the effective key size by half. However, AES-256 is still vulnerable to side-channel attacks, such as timing attacks or power analysis, because these attacks exploit physical implementations of the algorithm, not the algorithm’s cryptographic strength.

**Conclusion:** *The lab provided a deeper understanding of the AES encryption algorithm, its steps, and key expansion process. It highlighted the importance of AES's security in modern cryptography, though it also revealed potential vulnerabilities such as susceptibility to side-channel attacks.*