ECE 592: Assignment 1- Implementing AES

Nikshipth Maddugaru ‖ *nmaddug@ncsu.edu* ‖ 200511463

September 8, 2024

1 Introduction

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm established by the National Institute of Standards and Technology (NIST) in 2001. It is designed to replace the Data Encryption Standard (DES), offering enhanced security and efficiency. AES operates on 128-bit blocks of data and supports key sizes of 128, 192, or 256 bits, making it suitable for a wide range of applications. This report explores the implementation of AES in C++, detailing its core components and operations, which ensure robust encryption and decryption of data.

2 Implementation

Class Definition and Initialization

1. **S-Box and Inverse S-Box**
   - **S-Box**: A substitution box used in AES for byte substitution during encryption.
   - **InvSBox**: The inverse S-Box used during decryption.
2. **MixColumns and Inverse MixColumns**
   - **MixColumns**: A matrix multiplication operation used to mix the bytes of the state in encryption.
   - **InvMixColumns**: The inverse of the MixColumns matrix used during decryption.
3. **AES Class**
   - **Attributes**:
     - `input`: The input data in hexadecimal - string format.
     - `keyArray`: Array of round keys used for encryption and decryption.
     - `DecKeyArray`: Array of round keys used during decryption.
     - `addRounds`, `sBoxArray`, `shiftArray`, `mixColArray`: Intermediate matrices for various AES transformations.
   - **Methods**:
     - `init()`: Initializes key arrays and input matrix.
     - `inpToMatrix(string input)`: Converts input hexadecimal string to matrix format.
     - `sBoxSubstitution(int roundNo, int inputArr[4][4])`: Applies the S-Box substitution.
     - `shiftRows(int roundNo, int array[4][4])`: Performs row shifting.
     - `mixColumns(int roundNo, int array[4][4])`: Applies the MixColumns transformation.
     - `addRoundKey(int roundNo, int array[4][4])`: Applies the AddRoundKey transformation.
     - `encryption()`: Handles AES encryption process.
     - `decryption()`: Handles AES decryption process.

Step-by-Step Encryption Process

1. **Initial AddRoundKey**: The encryption process begins with the `addRoundKey` function, which XORs the input matrix with the initial round key (`keyArray[0]`). This initial transformation is stored in `addRounds[0]`.

2. **Encryption Rounds**: The code then enters a loop that runs for 10 rounds (0 to 10). Each round consists of the following steps:

   (a) **SubBytes**: The `sBoxSubstitution()` function replaces each byte in the state matrix with the corresponding byte from the S-Box, resulting in `sBoxArray[roundNo]`.

   (b) **ShiftRows**: The `shiftRows()` function shifts the rows of the state matrix by different offsets, resulting in `shiftArray[roundNo]`.

   (c) **MixColumns**: The `mixColumns()` function mixes the columns of the state matrix using matrix multiplication in GF(2^8, producing `mixColArray[roundNo]`. This step is skipped in the final round (Round 10).

   (d) **AddRoundKey**: The `addRoundKey()` function XORs the state matrix with the current round key (`keyArray[roundNo]`), resulting in `addRounds[roundNo]`.

3. **Final Round**: In the final round (Round 10), the `MixColumns` step is omitted, and the state matrix is directly XORed with the final round key to produce the final encrypted output.

   The decryption process reverses these steps, using the inverse operations and round keys to recover the original plaintext.

Step-by-Step Decryption Process The decryption process in AES is essentially the reverse of the encryption process, using the inverse transformations to recover the original plaintext. The following steps outline the decryption process:

(a) **Initial AddRoundKey**:
The decryption process begins by applying the `addRoundKey` function, XORing the ciphertext with the final round key (`DecKeyArray[10]`). This reverses the final round key addition in encryption and is stored in `addRounds[10]`.

(b) **Decryption Rounds**:
The decryption process then proceeds through 10 rounds (starting from Round 9 down to Round 0). Each round consists of the following steps in reverse order compared to encryption:

- **InvShiftRows**:
  The `invShiftRows()` function shifts the rows of the state matrix in the reverse direction of the encryption process. This repositions the bytes within the matrix to their original locations.
- **InvSubBytes**:
  The `invSBoxSubstitution()` function applies the inverse S-Box substitution, replacing each byte in the state matrix with its corresponding inverse from the InvSBox table.

- **AddRoundKey**:
  The `addRoundKey()` function XORs the state matrix with the round key for the current round (`DecKeyArray[roundNo]`). This undoes the key addition performed during encryption.
- **InvMixColumns**:
  In rounds 1 through 9, the `invMixColumns()` function reverses the MixColumns transformation by performing the inverse matrix multiplication in the Galois Field ($GF(2\hat{8})$). This restores the mixed columns to their original values before the encryption process.

(c) **Final Round**:
In the final round (Round 0), the `InvMixColumns` step is skipped. Instead, after applying the `InvShiftRows` and `InvSubBytes` operations, the state matrix is XORed with the initial round key (`DecKeyArray[0]`), producing the final decrypted output.

3 Assignment Questions

Input: 00112233445566778899aabbccddeeff

Question 1: What is the value you obtain? This is the value (round[1].start) you start your first round of AES.
Ans. 000102030405060708090a0b0c0d0e0f0

Question 2: What is the value (round[1].s_box) you obtain?
Ans. 63cab7040953d051cd60e0e7ba70e18c

Question 3: What is the matrix you obtain?
Ans.
63 09 cd ba
ca 53 60 70
b7 d0 e0 e1
04 51 e7 8c

Question 4: What is the value (round[1].s_row) you obtain?
Ans. 6353e08c0960e104cd70b751bacad0e7

Question 5: What is the value (round[1].m_col) you obtain?
Ans. 5f72641557f5bc92f7be3b291db9f91a

Question 6: What is the value (round[2].start) you obtain?
Ans. 89d810e8855ace682d1843d8cb128fe4

Question 7: What is the value (round[10].start) you obtain?
Ans. bd6e7c3df2b5779e0b61216e8b10b689

Question 8: What is the output ciphertext you obtain?
Ans. 69c4e0d86a7b0430d8cdb78070b4c55a

Question 9: Report how much time you spent on this assignment in hours.
Ans. About 6-7 Hours

BONUS Attempted?
Ans. Implemented Decryption.