Assignment 1: Implementing AES

Author: Venkatesh Rajagopalan

September 7, 2024

1 Introduction

This assignment aims to implement the AES encryption and decryption algorithm in C++. The objective is to understand the core operations involved in AES, including SubBytes, ShiftRows, MixColumns and AddRoundKey. The assignment will also involve verifying the correctness of the AES implementation using a specific plaintext/ciphertext along with a Master Key.

NOTE : This assignment includes code components that were partially generated with the assistance of ChatGPT. Specifically, the MixColumns transformation was implemented with guidance on the multiplication process within the Galois Field GF(2^8).

2 Implementation

2.1 SubBytes

The SubBytes operation is a non-linear substitution step where each byte in the state array is replaced with its corresponding value from the AES S-Box. This substitution is applied independently to each byte, enhancing the confusion properties of the encryption process by increasing the complexity of the relationship between the plaintext and the ciphertext.

2.2 ShiftRows

The ShiftRows operation involves rotating the rows of the state array to the left by varying offsets. Specifically, the first row remains unchanged, the second row is shifted by one byte, the third row by two bytes, and the fourth row by three bytes. This step contributes to the diffusion of the plaintext across multiple columns, complicating the encryption process.

2.3 MixColumns

The MixColumns operation transforms each column of the state array by multiplying it with a fixed polynomial in the Galois Field GF(2^8). This step combines the bytes within each column, further diffusing the data and ensuring that changes to one byte affect multiple bytes in subsequent rounds.

2.4 AddRoundKey

The AddRoundKey operation involves XORing each byte of the state array with the corresponding byte of the round key. This step is crucial as it combines the plaintext with the key material, ensuring that the encryption is dependent on the secret key. It is performed in every round of AES, including the initial and final rounds.

3 AES Encryption

In this section, the implemented AES encryption algorithm is tested using a predefined plaintext and a Master Key. The inputs are in hexadecimal format.

plaintext = '00112233445566778899aabbccddeeff'
secret key = '000102030405060708090a0b0c0d0e0f'

3.1 Test

Q1: What is the value (round[1].start) you obtain?
Answer: Round[1].start: 00102030405060708090a0b0c0d0e0f0.

Q2: What is the value (round[1].s_box) you obtain?
Answer: Round[1].s_box: 63cab7040953d051cd60e0e7ba70e18c.

Q3: What is the matrix you obtain?
Answer:

$$\begin{pmatrix} 63 & ca & b7 & 04 \\ 09 & 53 & d0 & 51 \\ cd & 60 & e0 & e7 \\ ba & 70 & e1 & 8c \end{pmatrix}$$

Q4: What is the value (round[1].s_row) you obtain?
Answer:

$$\begin{pmatrix} 63 & 53 & e0 & 8c \\ 09 & 60 & e1 & 04 \\ cd & 70 & b7 & 51 \\ ba & ca & d0 & e7 \end{pmatrix}$$

Q5: What is the value (round[1].m_col) you obtain?
Answer:

$$\begin{pmatrix} 5f & 72 & 64 & 15 \\ 57 & f5 & bc & 92 \\ f7 & be & 3b & 29 \\ 1d & b9 & f9 & 1a \end{pmatrix}$$

Q6: What is the value (round[2].start) you obtain?
Answer: Round[2].start: 89d810e8855ace682d1843d8cb128fe4

Q7: What is the value (round[10].start) you obtain?
Answer: Round[10].start: bd6e7c3df2b5779e0b61216e8b10b689

Q8: What is the output ciphertext you obtain?
Answer: ciphertext: 69c4e0d86a7b0430d8cdb78070b4c55a

Q9: Report how much time you spent on this assignment in hours.
Answer: The entire assignment where I implemented both the encryption and decryption algorithm took me about 16 hours.

4 Simulation

Commands to simulate the implementation

1. AES_ENC_CPP:

```
g++ -o AES_ENC sbox.cpp subbytes.cpp shiftrows.cpp
mixcolumns.cpp addroundkey.cpp AES_ENC.cpp

./AES_ENC > AES_ENC_128_out.txt
```

2. AES_DEC_CPP:

```
g++ -o AES_DEC inv_sbox.cpp inv_subbytes.cpp
inv_shiftrows.cpp inv_mixcolumns.cpp addroundkey.cpp AES_DEC.cpp

./AES_DEC > AES_DEC_128_out.txt
```

5 Conclusion

Through this assignment, the implementation of the AES encryption and decryption algorithms in C++ was successfully completed and verified. Key lessons learned include a deeper understanding of the AES algorithm, particularly the MixColumns transformation, as well as the importance of correctly handling bitwise operations. The assignment also highlighted the significance of thorough testing and verification using predefined test vectors. Future improvements could involve optimizing the implementation for speed and efficiency, as well as exploring additional enhancements for both encryption and decryption processes.