Topics in Cryptanalysis (CSE653)
Assignment no. 1
(Submitted By Vansh, 2021363)

Programming language: C++
Platform: MS Windows

In this assignment, we are implementing a AES (Advanced Encryption Standard) which uses a symmetric encryption technique in 128 bits of fixed-size blocks used to encrypt data
the encryption process uses 10 rounds of operations, where each round consist of four steps - SubBytes, ShiftRows, MixColumns, and AddRoundKey
Every byte in the block is changed in the SubBytes step by matching byte from a predetermined substitution table known as S-box. The block is moved one byte to the left in the second row, two bytes to the left in the third row, and three bytes to the left in the fourth row during the ShiftRows step. Every column in the block is multiplied in the MixColumns step.
to guarantee that the diffusion process has an impact on every byte in the block using a fixed matrix. Lastly, a key schedule is used to XOR the block with a subset of the key that is obtained from the main key in the AddRoundKey phase.
We defined the Original key as a 4x4 matrix of hexadecimal strings and constructed the Sbox which is used as the substitution tables in the encryption process, from the original key (which is of size 128 bits or 16 bytes) generating subkeys which are stored in the sub_key array during key expansion process to be used in each of the rounds(total of 10 rounds) of AES Algorithms, then for the AES encryption process by taking 4 plain texts as input and similarly AES Decryption using Inverse operations of the cipher text that is produced, both the plain text of the first encryption and decryption of the 9th cipher text matches and vice-versa

<u>Functionalities implemented in our system</u>:

- Construct_s_box: Constructs the s_box in the form of a C++ map.
- Construct_inverse_s_box: Constructs the inverse_s_box in the form of a C++ map.
- Copy_data: Copy data of one matrix into another.
- Hex_string_to_binary: Converts a string of 2 hexadecimal digits into the binary representation stored inside a string.
- Binary_to_hex: Converts a string of 8 size representing 8 bits into a corresponding string of 2 hexadecimal digits.
- Xor_operation: perform XOR operation on 4 bytes

.

- GF_multiplication: perform Galois Field multiplication of 2 strings representing 2 hexadecimal digits each.
- Substitute_bytes: perform substitute bytes transformation of AES
- Inverse_sub_bytes: perform Inverse substitution byte transformation of AES
- Shift_rows: perform shift rows transformation of AES
- Inverse_shift_rows: perform inverse transformation of AES
- Mix_column: perform mix columns transformation of AES
- Inverse_mix_column: perform inverse mix columns transformation of AES
- Add_round_key: perform XOR operation with the respective sub-key of that round.
- G_function: perform shift_rows, then substitute byte, then xor with a special value to obtain a value.
- Generate_sub_key: function used to generate 10 subkeys for each round.
- Matrix_to_string: converts a matrix with every element being 2 hexadecimal digits stored as a string to a string of hexadecimals.
- String_to_matrix: converts a string of hexadecimals into a matrix with 2 hexadecimals each in every element.
- AES_Encryption: perform AES encryption for a given plaintext.
- AES_Decryption: perform AES decryption for a given ciphertext.
- Generate_plaintext: convert a string of characters(1 byte) into a string of corresponding hexadecimals(4 bit)
- Generate_originaltext: convert a string of hexadecimals into a string of characters(1 byte).

Instructions to run code:

On any terminal type g++ Vansh_2021363.cpp, press enter.
Then, type ./a.out to execute the program.
Give inputs in the terminal like plaintext, key, and number of rounds.

Optimizations:

1. **Used Multi-threading to parallelize operations.**
2. **Used unordered_map** for faster lookups in S-box and inverseSbox O(1) average time.
3. **Parallelize Individual Rounds/Steps (Single-Block Parallelism):**
   A. **SubBytes:** Each of the 16 bytes in the state can be substituted independently, i.e. S-Box Lookup. This means we can do 16 lookups in parallel.
   B. **ShiftRows:** Shiftrows is mostly just re-indexing, Each row can be processed in parallel as they don't depend on each other.
   C. **MixColumns:** Each column can be mixed independently of the others. In other words, column 0 does not depend on column 1, 2, or 3 until after the operation. We can parallelize the mixing of the four columns.
   D. **AddRoundKey:** Each column is XORed with part of the round key. Each byte is independent and can be processed in parallel.
4. **Parallelize at the Block level(Multi-Block Parallelism):**
   If we have a plaintext of more than 128 bits, we can divide it into blocks of size 128 bits and carry out encryption, decryption of these individual blocks separately.
5. **Pre-Compute the sub-keys** at the starting and store them to be used again in Decryption algorithm after encryption algorithm.
6. Construction of S-Box, Inverse S-Box, Generation of sub keys, taking input can be done in parallel to each other.

Sample Input:

```
Enter Plaintext (16 characters): Whatsapp*Youtube
Enter Key (16 characters): Thats my Kung Fu
Enter Number of Rounds: 10

Plaintext(in hexadecimals)576861747361707 02A596F7574756265

Key(in hexadecimals)5468617473206D79204B756E67204675
```

```
Encrypted Plaintext(in hexadecimals) 8EAD37C0E8514988E60976DD0CC6B4DE

Encrypted plaintext(in characters) with key (in hexadecimal) "Thats my Kung Fu" in 10 rounds: Ä¡7└ÐQIêµ v
```

```
Decrypted Ciphertext(in hexadecimals) 57686174736170702A596F7574756265

Decrypted ciphertext(in characters) with key(in hexadecimals) "Thats my Kung Fu" in 10 rounds: Whatsapp*Youtube
```