Cryptography, Network and Security

Assignment 6

Apply AES algorithm for practical applications

Code:

```
#include <iostream>
#include <iomanip>
#include <cstdint>

using namespace std;

const int Nb = 4;
const int Nk = 4;
const int Nr = 10;

uint8_t s_box[256] = {

    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};

// Rijndael's Galois field multiplication for MixColumns
uint8_t gmul(uint8_t a, uint8_t b)
{
    uint8_t p = 0;
    uint8_t hi_bit_set;
    for (int i = 0; i < 8; i++)
    {
        if (b & 1)
        {
            p ^= a;
        }
        hi_bit_set = a & 0x80;
        a <<= 1;
        if (hi_bit_set)
        {
            a ^= 0x1b;
        }
        b >>= 1;
    }
    return p;
}

// Function to substitute bytes using the S-box
void subBytes(uint8_t state[4][4])
```

```
{
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            state[i][j] = s_box[state[i][j]];
        }
    }
}

// Function to shift rows in the state
void shiftRows(uint8_t state[4][4])
{
    uint8_t temp;

    temp = state[1][0];
    for (int i = 0; i < 3; i++)
    {
        state[1][i] = state[1][i + 1];
    }
    state[1][3] = temp;

    temp = state[2][0];
    state[2][0] = state[2][2];
    state[2][2] = temp;
    temp = state[2][1];
    state[2][1] = state[2][3];
    state[2][3] = temp;

    temp = state[3][3];
    for (int i = 3; i > 0; i--)
    {
        state[3][i] = state[3][i - 1];
    }
    state[3][0] = temp;
}

// Function to mix columns in the state matrix
void mixColumns(uint8_t state[4][4])
{
    uint8_t temp[4];
    for (int i = 0; i < 4; i++)
    {
        temp[0] = gmul(state[0][i], 2) ^ gmul(state[1][i], 3) ^
state[2][i] ^ state[3][i];
        temp[1] = state[0][i] ^ gmul(state[1][i], 2) ^ gmul(state[2][i],
3) ^ state[3][i];
```

```
        temp[2] = state[0][i] ^ state[1][i] ^ gmul(state[2][i], 2) ^
gmul(state[3][i], 3);
        temp[3] = gmul(state[0][i], 3) ^ state[1][i] ^ state[2][i] ^
gmul(state[3][i], 2);

        state[0][i] = temp[0];
        state[1][i] = temp[1];
        state[2][i] = temp[2];
        state[3][i] = temp[3];
    }
}

// Function to add round key to state
void addRoundKey(uint8_t state[4][4], uint8_t roundKey[4][4])
{
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            state[i][j] ^= roundKey[i][j];
        }
    }
}

// Simplified key expansion and encryption example
void AES_encrypt(uint8_t input[16], uint8_t key[16])
{
    uint8_t state[4][4];
    uint8_t roundKey[4][4];

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            state[j][i] = input[i * 4 + j];
        }
    }

    for (int i = 0; i < 16; i++)
    {
        roundKey[i % 4][i / 4] = key[i];
    }
    addRoundKey(state, roundKey);

    for (int round = 0; round < 9; round++)
    {
        subBytes(state);
        shiftRows(state);
```

```
        mixColumns(state);
        addRoundKey(state, roundKey);
    }

    subBytes(state);
    shiftRows(state);
    addRoundKey(state, roundKey);

    cout << "Ciphertext: ";
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            cout << hex << setw(2) << setfill('0') << (int)state[j][i] <<
" ";
        }
    }
    cout << endl;
}

int main()
{
    uint8_t plaintext[16] = {0x32, 0x43, 0xf6, 0xa8, 0x88, 0x5a, 0x30,
0x8d, 0x31, 0x31, 0x98, 0xa2, 0xe0, 0x37, 0x07, 0x34};
    uint8_t key[16] = {0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6,
0xab, 0xf7, 0xcf, 0x15, 0x88, 0x09, 0xcf, 0x4f};

    AES_encrypt(plaintext, key);

    return 0;
}
```