EXP NO-6

#include <stdio.h>

#include <stdint.h>

#define AES\_BLOCK\_SIZE 16

// AES S-box

static const uint8\_t s\_box[256] = {

// S-box values

};

// AES inverse S-box

static const uint8\_t inv\_s\_box[256] = {

// Inverse S-box values

};

// AES Rcon

static const uint8\_t rcon[10] = {

// Rcon values

};

// Multiplication in GF(2^8) with the AES polynomial x^8 + x^4 + x^3 + x + 1

uint8\_t aes\_mul(uint8\_t a, uint8\_t b) {

uint8\_t result = 0;

while (b) {

if (b & 1) {

result ^= a;

}

if (a & 0x80) {

a = (a << 1) ^ 0x1B; // xtime(a) with the AES polynomial

} else {

a <<= 1;

}

b >>= 1;

}

return result;

}

// SubBytes: substitute each byte of the state matrix with the value in the S-box

void SubBytes(uint8\_t \*state) {

for (int i = 0; i < AES\_BLOCK\_SIZE; i++) {

state[i] = s\_box[state[i]];

}

}

// ShiftRows: cyclically shift the rows of the state matrix

void ShiftRows(uint8\_t \*state) {

// Second row shift

uint8\_t temp = state[1];

state[1] = state[5];

state[5] = state[9];

state[9] = state[13];

state[13] = temp;

// Third row shift

temp = state[2];

state[2] = state[10];

state[10] = temp;

temp = state[6];

state[6] = state[14];

state[14] = temp;

// Fourth row shift

temp = state[3];

state[3] = state[15];

state[15] = state[11];

state[11] = state[7];

state[7] = temp;

}

// MixColumns: perform matrix multiplication on each column of the state matrix

void MixColumns(uint8\_t \*state) {

for (int i = 0; i < 4; i++) {

uint8\_t s0 = state[i];

uint8\_t s1 = state[i + 4];

uint8\_t s2 = state[i + 8];

uint8\_t s3 = state[i + 12];

state[i] = aes\_mul(0x02, s0) ^ aes\_mul(0x03, s1) ^ s2 ^ s3;

state[i + 4] = s0 ^ aes\_mul(0x02, s1) ^ aes\_mul(0x03, s2) ^ s3;

state[i + 8] = s0 ^ s1 ^ aes\_mul(0x02, s2) ^ aes\_mul(0x03, s3);

state[i + 12] = aes\_mul(0x03, s0) ^ s1 ^ s2 ^ aes\_mul(0x02, s3);

}

}

// AddRoundKey: XOR the state matrix with the round key

void AddRoundKey(uint8\_t \*state, const uint8\_t \*roundKey) {

for (int i = 0; i < AES\_BLOCK\_SIZE; i++) {

state[i] ^= roundKey[i];

}

}

int main() {

// Example usage of AES modules

uint8\_t state[AES\_BLOCK\_SIZE] = {0x32, 0x88, 0x31, 0xe0, 0x43, 0x5a, 0x31, 0x37, 0xf6, 0x30, 0x98, 0x07, 0xa8, 0x8d, 0xa2, 0x34};

uint8\_t roundKey[AES\_BLOCK\_SIZE] = {0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6, 0xab, 0xf7, 0x97, 0x76, 0x46, 0x6a, 0x94, 0x4c};

// Step 1: SubBytes

SubBytes(state);

// Step 2: ShiftRows

ShiftRows(state);

// Step 3: MixColumns

MixColumns(state);

// Step 4: AddRoundKey

AddRoundKey(state, roundKey);

// Display the state matrix after applying the AES operations

printf("State Matrix after AES operations:\n");

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

printf("%02x ", state[i + 4 \* j]);

}

printf("\n");

}

return 0;

}

