/\*

Anudish Jinturkar

H-27

Subject: Network Security (Lab)

Lab 7: Implementation of S-AES algorithm

\*/

#include <bits/stdc++.h>

#define gc getchar\_unlocked

using namespace std;

int w0[8],w1[8],w[8],arr[10],w3[8],w2[8],w4[8],w5[8],key1[16],key2[16],res[16],s00[4],s01[4],s10[4],s11[4],s\_arr1[16],cipher[16];

int binary[4]; int dummy1[8]; int dummy2[8];

int\* binary1; int\* binary2;

string plaintext,key;

int\* temp1; int\* temp2; int\* temp3; int\* temp4 ;

int\* tmp1; int\* tmp2; int\* tmp3; int\* tmp4 ;

int temp11[4],temp12[4],temp13[4],temp14[4];

int tmp11[4],tmp12[4],tmp13[4],tmp14[4];

int encryption\_sbox[] = {9,4,10,11,13,1,8,5,6,2,0,3,12,14,15,7};

int decription\_sbox[] = {10,5,9,11,1,7,8,15,6,0,2,3,12,4,13,14};

int s1[] = {1,0,0,0,0,0,0,0};

int s2[] = {0,0,1,1,0,0,0,0};

int multiplication\_1[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

int multiplication\_4[] = {0,4,8,12,3,7,11,15,6,2,14,10,5,1,13,9};

int multiplication\_2[] = {0,2,4,6,8,10,12,14,3,1,7,5,11,9,15,13};

int multiplication\_9[] = {0,9,1,8,2,11,3,10,4,13,5,12,6,15,7,14};

int encryption\_matrix[] = {1,4,4,1};

int decription\_matrix[] = {9,2,2,9};

int convert\_to\_integer(int nibble[]){

int x;

x = 1\*nibble[3] + 2\*nibble[2] + 4\*nibble[1] + 8\*nibble[0];

return x;

}

int\* convert\_to\_binary(int a){

int p=0;

int reminder;

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

arr[i] = 0;

while(a>0){

reminder = a%2;

a = a/2;

arr[p] = reminder;

p++;

}

int j=0;

for(int i=3; i>=0; i--){

binary[j] = arr[i];

j++;

}

return binary;

}

int\* rotateNibble(int w[]){

for(int i=0;i<8;i++){

if(i<4)

dummy1[i+4] = w[i];

else

dummy1[i-4] = w[i];

}

return dummy1;

}

int\* substituteNibble(int rw[],int sbx[]){

int first\_nibble[4]; int second\_nibble[4];

int n1,n2;

for(int i=0;i<8;i++){

if(i<4)

first\_nibble[i] = rw[i];

else

second\_nibble[i-4] = rw[i];

}

n1 = convert\_to\_integer(first\_nibble);

n2 = convert\_to\_integer(second\_nibble);

binary1 = convert\_to\_binary(sbx[n1]);

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

dummy2[i] = binary1[i];

}

binary2 = convert\_to\_binary(sbx[n2]);

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

dummy2[i+4] = binary2[i];

}

return dummy2;

}

int\* calculate\_w2(int w0[],int w1[],int s[]){

int wi[8]; int wj[8]; int\* rotate; int\* substitute;

rotate = rotateNibble(w1);

for(int i=0;i<8;i++)

wi[i] = rotate[i];

substitute = substituteNibble(wi,encryption\_sbox);

for(int i=0;i<8;i++)

wj[i] = substitute[i];

for(int i=0;i<8;i++){

wi[i] = w0[i] ^ s[i];

w[i] = wi[i] ^ wj[i];

}

return w;

}

int\* addRoundKey(){

cout<<"\n\n"<<" addRoundKey result: ";

for(int i=0;i<16;i++){

res[i] = plaintext[i] ^ key[i];

cout<<res[i];

}

return res;

}

int\* MixColumn(int arr[], int GF\_mul[],int multiplication\_1[]){

int arr\_nib1[4],arr\_nib2[4],arr\_nib3[4],arr\_nib4[4];

int p,q,r,s;

for(int i=0;i<16;i++){

if(i<4)

arr\_nib1[i] = arr[i];

else if(i>3 && i<8)

arr\_nib2[i-4] = arr[i];

else if(i>7 && i<12)

arr\_nib3[i-8] = arr[i];

else

arr\_nib4[i-12] = arr[i];

}

p = convert\_to\_integer(arr\_nib1);

q = convert\_to\_integer(arr\_nib2);

r = convert\_to\_integer(arr\_nib3);

s = convert\_to\_integer(arr\_nib4);

temp1 = convert\_to\_binary(GF\_mul[r]);

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

temp11[i] = temp1[i];

tmp4 = convert\_to\_binary(multiplication\_1[s]);

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

tmp14[i] = tmp4[i];

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

s11[i] = temp11[i] ^ tmp14[i];

}

temp2 = convert\_to\_binary(GF\_mul[q]);

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

temp12[i] = temp2[i];

tmp2 = convert\_to\_binary(multiplication\_1[p]);

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

tmp12[i] = tmp2[i];

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

s00[i] = tmp12[i] ^ temp12[i];

}

temp3 = convert\_to\_binary(GF\_mul[p]);

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

temp13[i] = temp3[i];

tmp3 = convert\_to\_binary(multiplication\_1[q]);

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

tmp13[i] = tmp3[i];

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

s10[i] = tmp13[i] ^ temp13[i];

}

temp4 = convert\_to\_binary(GF\_mul[s]);

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

temp14[i] = temp4[i];

tmp1 = convert\_to\_binary(multiplication\_1[r]);

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

tmp11[i] = tmp1[i];

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

s01[i] = tmp11[i] ^ temp14[i];

}

cout<<"\n\n"<<"S00 : ";

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

cout<<s00[i];

cout<<"\n\n"<<" s10 : ";

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

cout<<s10[i];

cout<<"\n\n"<<" s01 : ";

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

cout<<s01[i];

cout<<"\n\n"<<" s11 : ";

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

cout<<s11[i];

for(int i=0;i<16;i++){

if(i<4)

s\_arr1[i] = s00[i];

else if(i>3 && i<8)

s\_arr1[i] = s10[i-4];

else if(i>7 && i<12)

s\_arr1[i] = s01[i-8];

else

s\_arr1[i] = s11[i-12];

}

cout<<"\n\n"<<"S\_array : ";

for(int i=0;i<16;i++)

cout<<s\_arr1[i];

return s\_arr1;

}

void key\_calculation()

{

for(int i=0;i<16;i++){

if(i<8)

w0[i] = key[i]-48;

else

w1[i-8] = key[i]-48;

}

cout<<"\n\n"<<"W0 : ";

for(int i=0;i<8;i++)

cout<<w0[i];

cout<<"\n\n"<<"W1 : ";

for(int i=0;i<8;i++)

cout<<w1[i];

int\* temp\_w2;

temp\_w2 = calculate\_w2(w0,w1,s1);

cout<<"\n\n"<<"W2 : ";

for(int i=0;i<8;i++){

w2[i] = temp\_w2[i];

cout<<w2[i];

}

cout<<"\n\n"<<"W3 : ";

for(int i=0;i<8;i++){

w3[i] = w2[i] ^ w1[i];

cout<<w3[i];

}

int\* temp\_w4;

temp\_w4 = calculate\_w2(w2,w3,s2);

cout<<"\n\n"<<"W4 : ";

for(int i=0;i<8;i++){

w4[i] = temp\_w4[i];

cout<<w4[i];

}

cout<<"\n\n"<<"W5 : ";

for(int i=0;i<8;i++){

w5[i] = w4[i] ^ w3[i];

cout<<w5[i];

}

for(int i=0;i<16;i++){

if(i<8){

key1[i] = w2[i];

key2[i] = w4[i];

}

else{

key1[i] = w3[i-8];

key2[i] = w5[i-8];

}

}

cout<<"\n\n"<<"Key1 : ";

for(int j=0;j<16;j++)

cout<<key1[j];

cout<<"\n\n"<<"Key2 : ";

for(int j=0;j<16;j++)

cout<<key2[j];

}

void encryption()

{

int\* result;

result = addRoundKey();

int division1[8],division2[8];

//S-box

for(int i=0;i<16;i++){

if(i<8)

division1[i] = result[i];

else

division2[i-8] = result[i];

}

int\* division1\_result; int\* division2\_result;

int division\_result\_combination[16];

division1\_result = substituteNibble(division1,encryption\_sbox);

cout<<"\n"<<"Result of division : ";

for(int i=0;i<8;i++){

cout<<division1\_result[i];

division\_result\_combination[i] = division1\_result[i];

}

division2\_result = substituteNibble(division2,encryption\_sbox);

for(int i=0;i<8;i++){

division\_result\_combination[i+8] = division2\_result[i];

}

//swap 2 and 4 nibble

int division\_result\_combination\_swapping[16];

for(int i=0;i<16;i++){

if(i<4)

division\_result\_combination\_swapping[i] = division\_result\_combination[i];

else if(i>3 && i<8)

division\_result\_combination\_swapping[i] = division\_result\_combination[i+8];

else if(i>7 && i<12)

division\_result\_combination\_swapping[i] = division\_result\_combination[i];

else

division\_result\_combination\_swapping[i] = division\_result\_combination[i-8];

}

cout<<"\n"<<"Result of division after swapping: ";

for(int i=0;i<16;i++)

cout<<division\_result\_combination\_swapping[i];

//Mix column

int\* s\_arr1\_result;

s\_arr1\_result = MixColumn(division\_result\_combination\_swapping,multiplication\_4,multiplication\_1);

//Add round key1

int xor\_result[16];

for(int j=0;j<16;j++)

xor\_result[j] = s\_arr1\_result[j] ^ key1[j];

//Final round (Encryption)

int xor\_result\_fhalf[8]; int xor\_result\_shalf[8];

for(int i=0;i<16;i++){

if(i<8)

xor\_result\_fhalf[i] = xor\_result[i];

else

xor\_result\_shalf[i-8] = xor\_result[i];

}

int\* xor\_result1; int\* xor\_result2;

int xor\_result\_combination[16];

xor\_result1 = substituteNibble(xor\_result\_fhalf,encryption\_sbox);

cout<<"\n"<<" xor result1 : ";

for(int i=0;i<8;i++){

cout<<xor\_result1[i];

xor\_result\_combination[i] = xor\_result1[i];

}

xor\_result2 = substituteNibble(xor\_result\_shalf,encryption\_sbox);

cout<<"\n"<<" xor\_result2 :";

for(int i=0;i<8;i++){

cout<<xor\_result2[i];

xor\_result\_combination[i+8] = xor\_result2[i];

}

//swap 2 and 4 nibble

int xor\_result\_combination\_swapping[16];

for(int i=0;i<16;i++){

if(i<4)

xor\_result\_combination\_swapping[i] = xor\_result\_combination[i];

else if(i>3 && i<8)

xor\_result\_combination\_swapping[i] = xor\_result\_combination[i+8];

else if(i>7 && i<12)

xor\_result\_combination\_swapping[i] = xor\_result\_combination[i];

else

xor\_result\_combination\_swapping[i] = xor\_result\_combination[i-8];

}

cout<<"\n"<<"xor\_result after swapping: ";

for(int i=0;i<16;i++)

cout<<xor\_result\_combination\_swapping[i];

//Add round key2

for(int j=0;j<16;j++)

cipher[j] = xor\_result\_combination\_swapping[j] ^ key2[j];

cout<<"\n\n"<<"Cipher Text is: ";

for(int k=0;k<16;k++)

cout<<cipher[k];

cout<<endl;

}

void decryption()

{

int decryption\_xor\_result[16];

//Add round key1

for(int j=0;j<16;j++)

decryption\_xor\_result[j] = cipher[j] ^ key2[j];

cout<<"\n"<<"Xor-result : ";

for(int i=0;i<16;i++)

cout<<decryption\_xor\_result[i];

//Final round (Encryption)

int decryption\_xor\_result\_fhalf[8]; int decryption\_xor\_result\_shalf[8];

for(int i=0;i<16;i++){

if(i<8)

decryption\_xor\_result\_fhalf[i] = decryption\_xor\_result[i];

else

decryption\_xor\_result\_shalf[i-8] = decryption\_xor\_result[i];

}

int\* dec\_xor\_result; int\* decryption\_xor\_result2;

int decryption\_xor\_result\_combine[16];

dec\_xor\_result = substituteNibble(decryption\_xor\_result\_fhalf,decription\_sbox);

cout<<"\n"<<" Decryption\_xor\_result : ";

for(int i=0;i<8;i++){

cout<<dec\_xor\_result[i];

decryption\_xor\_result\_combine[i] = dec\_xor\_result[i];

}

decryption\_xor\_result2 = substituteNibble(decryption\_xor\_result\_shalf,decription\_sbox);

for(int i=0;i<8;i++){

cout<<decryption\_xor\_result2[i];

decryption\_xor\_result\_combine[i+8] = decryption\_xor\_result2[i];

}

//swap 2 and 4 nibble

int dec\_xor\_result\_combination\_swapping[16];

for(int i=0;i<16;i++){

if(i<4)

dec\_xor\_result\_combination\_swapping[i] = decryption\_xor\_result\_combine[i];

else if(i>3 && i<8)

dec\_xor\_result\_combination\_swapping[i] = decryption\_xor\_result\_combine[i+8];

else if(i>7 && i<12)

dec\_xor\_result\_combination\_swapping[i] = decryption\_xor\_result\_combine[i];

else

dec\_xor\_result\_combination\_swapping[i] = decryption\_xor\_result\_combine[i-8];

}

cout<<"\n\n"<<"Decryption\_xor\_result after swapping : ";

for(int i=0;i<16;i++)

cout<<dec\_xor\_result\_combination\_swapping[i];

//Add round key1

int dec\_round1[16];

for(int j=0;j<16;j++)

dec\_round1[j] = dec\_xor\_result\_combination\_swapping[j] ^ key1[j];

cout<<"\n\n"<<" Decryption after round1 : ";

for(int k=0;k<16;k++)

cout<<dec\_round1[k];

int\* decryption\_matrix\_result;

decryption\_matrix\_result = MixColumn(dec\_round1,multiplication\_2,multiplication\_9);

cout<<"\n\n"<<"Result of matrix multiplication: ";

for(int i=0;i<16;i++)

cout<<decryption\_matrix\_result[i];

//swap 2 and 4 nibble

int decryption\_matrix\_result\_swapping[16];

for(int i=0;i<16;i++){

if(i<4)

decryption\_matrix\_result\_swapping[i] = decryption\_matrix\_result[i];

else if(i>3 && i<8)

decryption\_matrix\_result\_swapping[i] = decryption\_matrix\_result[i+8];

else if(i>7 && i<12)

decryption\_matrix\_result\_swapping[i] = decryption\_matrix\_result[i];

else

decryption\_matrix\_result\_swapping[i] = decryption\_matrix\_result[i-8];

}

cout<<"\n"<<"Decryption matrix result after swapping: ";

for(int i=0;i<16;i++)

cout<<decryption\_matrix\_result\_swapping[i];

//Inverse Nibble sub

int decryption\_matrix\_result\_fhalf[8]; int decryption\_matrix\_result\_shalf[8];

for(int i=0;i<16;i++){

if(i<8)

decryption\_matrix\_result\_fhalf[i] = decryption\_matrix\_result\_swapping[i];

else

decryption\_matrix\_result\_shalf[i-8] = decryption\_matrix\_result\_swapping[i];

}

int\* dec\_mat\_result; int\* decryption\_matrix\_result2;

int decryption\_matrix\_result\_combine[16];

dec\_mat\_result = substituteNibble(decryption\_matrix\_result\_fhalf,decription\_sbox);

for(int i=0;i<8;i++){

decryption\_matrix\_result\_combine[i] = dec\_mat\_result[i];

}

decryption\_matrix\_result2 = substituteNibble(decryption\_matrix\_result\_shalf,decription\_sbox);

for(int i=0;i<8;i++){

decryption\_matrix\_result\_combine[i+8] = decryption\_matrix\_result2[i];

}

//Add round key0

int decryption\_text[16];

for(int j=0;j<16;j++)

decryption\_text[j] = (decryption\_matrix\_result\_combine[j] ^ key[j])-48;

cout<<"\n"<<"Decryption result is: ";

for(int k=0;k<16;k++)

cout<<decryption\_text[k];

}

int main(int argc,char \*\*argv)

{

cout<<"\n"<<"Enter the plaintext(16-bits in binary form): "<<endl;

cin>>plaintext;

cout<<"\n"<<"Enter the key(16-bits in binary form): "<<endl;

cin>>key;

cout<<"\n";

key\_calculation();

encryption();

decryption();

cout<<"\n\n";

return 0;

}

/\*

OUTPUT:

Enter the plaintext(16-bits in binary form):

1101001000111001

Enter the key(16-bits in binary form):

1000110011100001

W0 : 10001100

W1 : 11100001

W2 : 01000011

W3 : 10100010

W4 : 11010011

W5 : 01110001

Key1 : 0100001110100010

Key2 : 1101001101110001

addRoundKey result: 0101111011011000

Result of division : 00011111

Result of division after swapping: 0001011011101111

S00 : 1010

s10 : 0010

s01 : 0111

s11 : 0010

S\_array : 1010001001110010

xor result1 : 11110100

xor\_result2 :11101001

xor\_result after swapping: 1111100111100100

Cipher Text is: 0010101010010101

Xor-result : 1111100111100100

Decryption\_xor\_result : 1110000011010001

Decryption\_xor\_result after swapping : 1110000111010000

Decryption after round1 : 1010001001110010

S00 : 0001

s10 : 0110

s01 : 1110

s11 : 1111

S\_array : 0001011011101111

Result of matrix multiplication: 0001011011101111

Decryption matrix result after swapping: 0001111111100110

Decryption result is: 1101001000111001

\*/