DES DOCUMENTATION

DES algorithm: DES adalah pola dasar algoritma blok cipher-yang mengambil fixed-length bit string plaintext dan mengubahnya melalui serangkaian operasi menjadi bitstring ciphertext lain yang sama panjang. Dalam kasus DES, ukuran blok adalah 64 bit. DES juga menggunakan kunci untuk menyesuaikan transformasi, sehingga dekripsi yang konon hanya bisa dilakukan oleh orang-orang yang mengetahui kunci tertentu yang digunakan untuk mengenkripsi. Kuncinya seolah-olah terdiri dari 64 bit. Namun, hanya 56 di antaranya benar-benar digunakan oleh algoritma. Delapan bit digunakan semata-mata untuk memeriksa parity, dan setelah itu dibuang. Oleh karena itu panjang kunci efektif adalah 56 bit.

Documentation of code

#include<iostream>

#include<algorithm>

#include<math.h>

#include<stdio.h>

#include<cstdlib>

using namespace std;

* Header < Iostream > untuk kode melibakan proses input dan output
* Header < algorithm> mendefinisikan koleksi fungsi khusus dirancang untuk digunakan pada rentang elemen .
* Header math.h mendefinisikan berbagai fungsi matematika dan satu makro . Semua fungsi yang tersedia di library ini mengambil dua kali lipat sebagai argumen dan kembali ganda sebagai hasilnya .
* Header stdio.h mendefinisikan tiga jenis variabel , beberapa macro , dan berbagai fungsi untuk melakukan input dan output .
* Header cstdlib mendefinisikan beberapa fungsi tujuan umum , termasuk manajemen yang dinamis memori , random nomor generasi , komunikasi dengan lingkungan , aritmatika integer, mencari , memilah dan mengkonversi .

int shiftschedule[16]={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};

* Adalah deklarasi untuk shift keys pada round 1 sampai ke round 16

int ip[64]={

58,50,42,34,26,18,10,2,

60,52,44,36,28,20,12,4,

62,54,46,38,30,22,14,6,

64,46,48,40,32,24,16,8,

57,49,41,33,25,17,9,1,

59,51,43,35,27,19,11,3,

61,53,45,37,29,21,13,5,

63,55,47,39,31,23,15,7

};

* Adalah deklarasi untuk indeks permutation dengan key 64 bit

int inverseip[64]={

40,8,48,16,56,24,64,32,

39,7,47,15,55,23,63,31,

38,6,46,14,54,22,62,30,

37,5,45,13,53,21,61,29,

36,4,44,12,52,20,60,28,

35,3,43,11,51,19,59,27,

34,2,42,10,50,18,58,26,

33,1,41,9,49,17,57,25

};

* Adalah deklarasi untuk inverse indeks permutation dengan key 64 bit

int e\_table[48]={

32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

8, 9, 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1

};

* Adalah deklarasi untuk expansion indeks permutation table

int pc1[56]={

57, 49, 41, 33, 25, 17, 9,

1, 58, 50, 42, 34, 26, 18,

10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7, 62, 54, 46, 38, 30, 22,

14, 6, 61, 53, 45, 37, 29,

21, 13, 5, 28, 20, 12, 4

};

* Adalah deklarasi untuk permutation combination ke-1

int pc2[48]={

14, 17, 11, 24, 1, 5, 3, 28,

15, 6, 21, 10, 23, 19, 12, 4,

26, 8, 16, 7, 27, 20, 13, 2,

41, 52, 31, 37,47, 55, 30,40,

41, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32

};

* Adalah deklarasi untuk permutation combination ke-2

string sbox[4][16]={

"1110", "0100", "1101", "0001", "0010", "1111", "1011", "1000", "0011", "1010", "0110", "1100", "0101", "1001", "0000", "0111",

"0000", "1111", "0111", "0100", "1110", "0010", "1101", "0001", "1010", "0110", "1100", "1011", "1001", "0101", "0011", "1000",

"0100", "0001", "1110", "1000", "1101", "0110", "0010", "1011", "1111", "1100", "1001", "0111", "0011", "1010", "0101", "0000",

"1111", "1100", "1000", "0010", "0100", "1001", "0001", "0111", "0101", "1011", "0011", "1110", "1010", "0000", "0110", "1101"

};

* Adalah deklarasi untuk substitution box

nt pbox[32]={

16,7,20,21,29,12,28,17,

1,15,23,26,5,18,31,10,

2,8,24,14,32,27,3,9,

19,13,30,6,22,11,4,25

};

* Adalah deklarasi untuk permutation box

string padding(string kalimat){

while(kalimat.length()%8!=0){

kalimat+='\*';

}

return kalimat;

}

* Adalah proses padding pada algoritma, yang menambahkan \* sebagai string tambahan yang berfungsi untuk menjadikan teks ke 64 bit

string converttobinary(int bilangan){

string s;

while(bilangan!=0){

if(bilangan%2==0){

s+="0";

}

else s+="1";

bilangan=bilangan/2;

}

while (s.length()!=8) s+="0";

string c = s;

for (int i=0;i<s.length();i++){

s[i]=c[s.length()-1-i];

}

return s;

}

* Adalah kode untuk mengubah setiap string ke dalam bentuk bilangan

int converttodec(string biner){

int n=0;

int c=biner.length()-1;

for(int i=0;i<biner.length();i++){

if(biner[i]=='1') n+=pow(2,c);

c--;

}

return n;

}

* Adalah kode untuk mengubah string ke dalam bentuk bilangan biner

void converttotext(string biner){

int n = biner.length();

int m = n/8;

string s[m];

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

string t;

t+=biner[i];t+=biner[i+1];t+=biner[i+2];t+=biner[i+3];

t+=biner[i+4];t+=biner[i+5];t+=biner[i+6];t+=biner[i+7];

s[i/8]=t;

// cout<<s[i/8]<<endl;

}

char u[m];

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

u[i]=converttodec(s[i]);

cout<<u[i]; }

}

* Adalah kode untuk mengubah bilangan biner ke dalam bentuk textt

string tobyte(string kata){

string s;

for(int i=0;i<kata.length();i++){

s+=converttobinary(kata[i]);

//s+=' ';

}

return s;

}

* Adalah kode untuk mengubah plaint text ke dalam bentuk byte

void split(string kata, string &kiri, string &kanan){

int n=kata.length()/2;

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

kiri+=kata[i];

kanan+=kata[i+n];

}

}

* Adalah fungsi untuk mengubah melakukan splitiing pada key pada round ke n

string leftshift(string key, int round){

int n = shiftschedule[round-1];

int m = key.length();

if(n==1){

char temp = key[0];

for(int j=0;j<m-1;j++){

key[j]=key[j+1];

}

key[m-1]=temp;

}

else if(n==2){

char temp1=key[0];

char temp2=key[1];

for(int j=0;j<m-2;j++){

key[j]=key[j+2];

}

key[m-2]=temp1;

key[m-1]=temp2;

}

return key;

}

* Adalah proses perlakuan left shift pada round ke n sebanyak x pada masing-masing key R dan L. dimana n x telah dideklarasikan pada shiftround

string merge(string kiri, string kanan){

return kiri + kanan;

}

string exor(string a, string b, int n){

string temp;

for(int i;i<n;i++){

if(a[i]==b[i]){

temp+="0";

}

else temp+="1";

}

return temp;

}

* Adalah proses perlakuan merging pada key L dan R
* Adalah proses exor pada operasi pada algoritma

string s\_box(string kata){

string s;

for(int i=0;i<48;i=i+6){

string a,b;

a=kata[i];a+=kata[i+5];

int a1 = converttodec(a);

b=kata[i+1];b+=kata[i+2];b+=kata[i+3];b+=kata[i+4];

int b1 = converttodec(b);

s+=sbox[a1][b1];

}

return s;

}

* Adalah proses subtitusi yang terjadi setelah proses perlakuan algoritma, yang melibatkans box yang sudah dideklarasi
* String text untuk masukkan
* String key untuk dijadikan sebagai key
* Adalah proses memasukkan text
* Adalah proses memasukkan ke
* Proses perngubahan text dan key ke byte
* Adalah peroses pengubahan text dan key dengan indeks permutation sebagai uturan
* Adalah proses DES diantaranya, ekspansion, splitting kry L dan R, proses shift left, merging l dan r.
* Adalah proses enkripsi pada plaintext ke chipper text

main(){

string text;

string key;

cout<<"Text(<=8 karakter): ";cin>>text;

cout<<"Key(8 karakter): ";cin>>key;

string s = tobyte(padding(text));

string s1= permutation(ip,64,s);

key = tobyte(key);

string k = permutation(pc1,56,key);

cout<<"Plaintext\t"<<s<<endl;

cout<<"I. Permuation\t"<<s1<<endl;

cout<<"Key\t\t"<<k<<endl<<endl;

string c, d;

split(k,c,d);

string l, r;

split(s1,l,r);

string m;

string roundkey[16];

cout<<"Enkripsi"<<endl;

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

cout<<"Round "<<i+1<<endl;

c=leftshift(c,i+1);

d=leftshift(d,i+1);

m=merge(c,d);

roundkey[i]=m;

cout<<"Round Key\t"<<m<<endl;

m=permutation(pc2,48,m);

cout<<"Permutation Key\t"<<m<<endl;

string temp = r;

r = permutation(e\_table,48,r);

cout<<"Expansion R \t"<<r<<endl;

r = exor(m,r,48);

cout<<"Xor R w/ Key\t"<<r<<endl;

r= s\_box(r);

cout<<"SBox R\t\t"<<r<<endl;

r= permutation(pbox,32,r);

cout<<"Pbox R\t\t"<<r<<endl;

r= exor(l,r,32);

cout<<"Xor R w/ L\t"<<r<<endl;

l= temp;

cout<<"Replace L\t"<<l<<endl;

cout<<"Round Result\t"<<l<<r<<endl<<endl;

}

string ciphertext = merge(l,r);

ciphertext=permutation(inverseip,64,ciphertext);

cout<<"Dekripsi"<<endl;

ciphertext=permutation(ip,64,ciphertext);

split(ciphertext,l,r);

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

cout<<"Round "<<i+1<<endl;

cout<<"Round Key\t"<<roundkey[15-i]<<endl;

m=permutation(pc2,48,roundkey[15-i]);

cout<<"Permutation Key\t"<<m<<endl;

string temp = l;

l = permutation(e\_table,48,l);

cout<<"Expansion L \t"<<r<<endl;

l = exor(m,l,48);

cout<<"Xor L w/ Key\t"<<r<<endl;

l = s\_box(l);

cout<<"SBox L\t\t"<<r<<endl;

l = permutation(pbox,32,l);

cout<<"Pbox L\t\t"<<r<<endl;

l = exor(l,r,32);

cout<<"Xor L w/ R\t"<<r<<endl;

r = temp;

cout<<"Replace R\t"<<r<<endl;

cout<<"Round Result\t"<<l<<r<<endl<<endl;

}

string plaintext= merge(l,r);

plaintext = permutation(inverseip,64,plaintext);

* Adalah proses deskripsi pada chipertext ke plaint text

Contoh Progam

Input: bahaya

Key: komputer

Text(<=8 karakter): bahaya

Key(8 karakter): komputer

Plaintext 0110001001100001011010000110000101111001011000010010101000101010

I. Permuation 0011111100010000000000000011101000000000111111111101010011000001

Key 00000000111111111111111110111000001101110110000001111000

Enkripsi

Round 1

Round Key 00000001111111111111111101100000011011101100000011110001

Permutation Key 111100001011111011101110110100001000011110011000

Expansion R 100000000001011111111111111010101001011000000010

Xor R w/ Key 011100001010100100010001001110100001000110011010

SBox R 00001111111010101000111100011001

Pbox R 01011011011010010110100101011100

Xor R w/ L 01100100011110010110100101100110

Replace L 00000000111111111101010011000001

Round Result 0000000011111111110101001100000101100100011110010110100101100110

Round 2

Round Key 00000011111111111111111011000000110111011000000111100010

Permutation Key 111000001011111011110110100101011011010010001100

Expansion R 001100001000001111110010101101010010101100001100

Xor R w/ Key 110100000011110100000100001000001001111110000000

SBox R 10011111100111010010111000001110

Pbox R 11011100101010100101000111111010

Xor R w/ L 11011100010101011000010100111011

Replace L 01100100011110010110100101100110

Round Result 0110010001111001011010010110011011011100010101011000010100111011

Round 3

Round Key 00001111111111111111101100000011011101100000011110001000

Permutation Key 111101001111111001110110001010000001011011100101

Expansion R 111011111000001010101011110000001010100111110111

Xor R w/ Key 000110110111110011011101111010001011111100010010

SBox R 00011110101100111010001001011010

Pbox R 11001111011110100000000110010110

Xor R w/ L 10101011000000110110100011110000

Replace L 11011100010101011000010100111011

Round Result 1101110001010101100001010011101110101011000000110110100011110000

. . .

Ciphertext ▼äu,|$╝╖

Dekripsi

Round 1

Round Key 00000000111111111111111110111000001101110110000001111000

Permutation Key 111100011011111000101110010000010000001001011110

Expansion L 1100001011111100010110011000000111000010111111000101100110000001

Xor L w/ Key 1100001011111100010110011000000111000010111111000101100110000001

SBox L 1100001011111100010110011000000111000010111111000101100110000001

Pbox L 1100001011111100010110011000000111000010111111000101100110000001

Xor L w/ R 1100001011111100010110011000000111000010111111000101100110000001

Replace R 0001010011010101111111111000010100010100110101011111111111000101

Round Result 0110011010010010111011110011101000010100110101011111111110000101

00010100110101011111111111000101

Round 2

Round Key 10000000011111111111111111010100000110111011000000111100

Permutation Key 111110011011111010100110110100111000101001000100

Expansion L 0001010011010101111111111000010100010100110101011111111111000101

Xor L w/ Key 0001010011010101111111111000010100010100110101011111111111000101

SBox L 0001010011010101111111111000010100010100110101011111111111000101

Pbox L 0001010011010101111111111000010100010100110101011111111111000101

Xor L w/ R 0001010011010101111111111000010100010100110101011111111111000101

Replace R 01100110100100101110111100111010

Round Result 1000100000011010011001101111010001100110100100101110111100111010

Round 3

Round Key 01100000000111111111111111110001000001101110110000001111

Permutation Key 110100111010111010101111100001001100001101110001

Expansion L 01100110100100101110111100111010

Xor L w/ Key 01100110100100101110111100111010

SBox L 01100110100100101110111100111010

Pbox L 01100110100100101110111100111010

Xor L w/ R 01100110100100101110111100111010

Replace R 10001000000110100110011011110100

Round Result 1000011000000010111100100000111010001000000110100110011011110100

. . .

plainttext bahaya\*\*

Plaint Text Awal =bahaya\*\*

Hasil enkripsi =▼äu,|$╝╖

Hasil dekripsi =bahaya\*\*