**DECRYPION**

void decryption(String file)

{

try

{

fdec=new DataOutputStream(new FileOutputStream("dec.txt"));

fenc1=new FileInputStream(file);

}

catch(Exception e)

{

System.out.println(e+"ravikrishna");

}

try

{

size=fenc1.available();

np=0;

p=size%8;

if(p!=0)

np=8-p;

size+=np;

if(p==0)

nl=size/8;

else

nl=size/8+1;

buf1=new byte[size+10];

buf=new byte[8];

fenc1.read(buf1);

mark=-8;

int arr[]=new int [8];

for(k1=0;k1<nl;k1++)

{

mark+=8;

for(int k2=0;k2<8;k2++)

buf[k2]=buf1[mark+k2];

for(int k2=0;k2<8;k2++)

{

arr[k2]=0;

if(buf[k2]>=0)

arr[k2]=buf[k2];

else

arr[k2]=buf[k2]+256;

}

j1=0;i1=0;

while( i1<=3)

{

if((j1+1)%2==0)

{

x[i1]|=arr[j1]; // dividing 64 bit cypher block into four 16 bit registers

i1++;

}

else

{

x[i1]=arr[j1];

x[i1]<<=8;

}

j1++;

}

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

{

ft1 =x[t]&255;

ft=x[t]>>8;

}

step1=mul( x[0] , minverse( z[48] ));

step2=( x[1] + ainverse( z[49] )) % 65536;

step3=( x[2] + ainverse( z[50] )) % 65536;

step4=mul( x[3] , minverse( z[51] ));

step5=step1^step3;

step6=step2^step4;

step7=mul(step5,z[46]);

step8=(step6+step7)%65536;

step9=mul(step8,z[47]);

step10=(step7+step9)%65536;

step11=step1^step9;

step12=step3^step9;

step13=step2^step10;

step14=step4^step10;

x[0]=step11;

x[1]=step12;

x[2]=step13;

x[3]=step14;

// 2nd round

int j2=40;

for(j1=1;j1<=7;j1++)

{

step1=mul( x[0] , minverse( z[j2+2] ));

step2=( x[1] + ainverse( z[j2+4] )) % 65536;

step3=( x[2] + ainverse( z[j2+3] )) % 65536;

t=step2;

step2=step3;

step3=t;

step4=mul( x[3] , minverse( z[j2+5] ));

step5=step1^step3;

step6=step2^step4;

step7=mul(step5,z[j2+0]);

step8=(step6+step7)%65536;

step9=mul(step8,z[j2+1]);

step10=(step7+step9)%65536;

step11=step1^step9;

step12=step3^step9;

step13=step2^step10;

step14=step4^step10;

x[0]=step11;

x[1]=step12;

x[2]=step13;

x[3]=step14;

j2=j2-6;

}

x[0]=mul(x[0],minverse(z[0]));

x[1]=(x[1]+ainverse(z[2]))%65536;

x[2]=(x[2]+ainverse(z[1]))%65536;

x[3]=mul(x[3],minverse(z[3]));

t=x[1];

x[1]=x[2];

x[2]=t;

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

{

ft1 =x[t]&255;

ft=x[t]>>8;

fdec.write((char)ft);

fdec.write((char)ft1);

}

}

}

catch(Exception e)

{

System.out.println(e+"ergeg");

}

}

int mul( int a , int b)

{

double c,d;

if (a==0)

c=65536;

if(b==0)

d=65536;

c=(double)a;

d=(double)b;

a= (int)((c\*d)%65537);

return a;

}

int minverse(int z)

{

int to,t1;

int q,y;

if(z<=1)

return z;

t1=0x10001/z;

y=0x10001%z;

if(y==1)

return (0xffff&(1-t1));

to=1;

do

{

q=z/y;

z=z%y;

to+=q\*t1;

if(z==1)

return to;

q=y/z;

y=y%z;

t1+=q\*to;

}

while(y!=1);

return (0xffff&(1-t1));

}

int ainverse(int z)

{

return (65536-z);

}

}

**ENCRYPTION**

void encrypt()

{

try

{

size=fin.available();

p=size%8;

if(p!=0)

np=8-p;

size+=np;

if(p==0)

nl=size/8;

else

nl=size/8+1;

buf=new byte[8];

buf1=new byte[size+10];

fin.read(buf1);

int enc[]=new int[size];

mark=-8;

k2=0;

for(k=0;k<nl;k++)

{

mark+=8;

for(int k1=0;k1<8;k1++)

buf[k1]=buf1[mark+k1];

i=0;

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

x[i]=0;

j1=0;i1=0;

while( i1<=3)

{

if((j1+1)%2==0)

{

x[i1]|=buf[j1]; // dividing 64 bit cypher block into four 16 bit registers

i1++;

}

else

{

x[i1]=buf[j1];

x[i1]<<=8;

}

j1++;

}

// 7 rounds and 14 steps

for(i=0 ; i <48 ; )

{

step1=mul(x[0],z[i+0]);

step2=(x[1]+z[i+1])%65536;

step3=(x[2]+z[i+2])%65536;

step4=mul(x[3],z[i+3]);

step5=step1^step3;

step6=step2^step4;

step7=mul(step5,z[i+4]);

step8=(step6+step7)%65536;

step9=mul(step8,z[i+5]);

step10=(step7+step9)%65536;

step11=step1^step9;

step12=step3^step9;

step13=step2^step10;

step14=step4^step10;

x[0]=step11;

x[1]=step13;

x[2]=step12;

x[3]=step14;

i=i+6;

}

x[0]=mul(x[0],z[48]);

x[1]=(x[1]+z[49])%65536;

x[2]=(x[2]+z[50])%65536;

x[3]=mul(x[3],z[51]);

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

{

ft1 =x[t]&255;

ft=x[t]>>8;

fenc.write((char)ft);

fenc.write((char)ft1);

}

}

fin.close();

}

catch(Exception e)

{

System.out.println(e);

}

}