**Exp 9: Implementation and analysis of RSA cryptosystem and RSA/ElGamel**

**RSA Algorithm**

1. Request two ***prime numbers*** from the user and verify them.
2. Put the prime numbers in different variables.
3. Determine ***n = pq***.
4. Calculate ***(n) = (p - 1)(q - 1)***after the above step.
5. Select a random number ***e*** that is close to being ***prime*** to both ***n*** and ***1 e n***.
6. Determine ***d = e-1 mod n***.
7. Print out the ***private*** and ***public keys***.
8. Request a message from the user and then save it in a variable.
9. Use the ***public key*** to encrypt the message.
10. Using the ***private key***, decrypt the message.
11. Print the ***message***, both ***encrypted*** and ***decrypted***.

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

void main() {

clrscr();

printf("\nENTER FIRST PRIME NUMBER\n");

scanf("%d",&p);

flag=prime(p);

if(flag==0) {

printf("\nWRONG INPUT\n");

getch();

exit(1);

}

printf("\nENTER ANOTHER PRIME NUMBER\n");

scanf("%d",&q);

flag=prime(q);

if(flag==0||p==q) {

printf("\nWRONG INPUT\n");

getch();

exit(1);

}

printf("\nENTER MESSAGE\n");

fflush(stdin);

scanf("%s",msg);

for (i=0;msg[i]!=NULL;i++)

m[i]=msg[i];

n=p\*q;

t=(p-1)\*(q-1);

ce();

printf("\nPOSSIBLE VALUES OF e AND d ARE\n");

for (i=0;i<j-1;i++)

printf("\n%ld\t%ld",e[i],d[i]);

encrypt();

decrypt();

getch();

}

int prime(long int pr) {

int i;

j=sqrt(pr);

for (i=2;i<=j;i++) {

if(pr%i==0)

return 0;

}

return 1;

}

void ce() {

int k;

k=0;

for (i=2;i<t;i++) {

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q) {

e[k]=i;

flag=cd(e[k]);

if(flag>0) {

d[k]=flag;

k++;

}

if(k==99)

break;

}

}

}

long int cd(long int x) {

long int k=1;

while(1) {

k=k+t;

if(k%x==0)

return(k/x);

}

}

void encrypt() {

long int pt,ct,key=e[0],k,len;

i=0;

len=strlen(msg);

while(i!=len) {

pt=m[i];

pt=pt-96;

k=1;

for (j=0;j<key;j++) {

k=k\*pt;

k=k%n;

}

temp[i]=k;

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\nTHE ENCRYPTED MESSAGE IS\n");

for (i=0;en[i]!=-1;i++)

printf("%c",en[i]);

}

void decrypt() {

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1) {

ct=temp[i];

k=1;

for (j=0;j<key;j++) {

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

m[i]=-1;

printf("\nTHE DECRYPTED MESSAGE IS\n");

for (i=0;m[i]!=-1;i++)

printf("%c",m[i]);

}

**Input & Output:**

1. ENTER FIRST PRIME NUMBER: 5
2. ENTER ANOTHER PRIME NUMBER: 17
3. ENTER MESSAGE: Jitendra
4. POSSIBLE VALUES OF e AND d ARE:
5. 5   65
6. 11  59
7. 13  25
8. 17  89
9. 23  47
10. 29  41
11. 31  7
12. 37  73
13. 41  29
14. THE ENCRYPTED MESSAGE IS:
15. I?j?x??a
16. THE DECRYPTED MESSAGE IS:
17. Jitendra

**Exp 10. Implementation of Diffie/Hellman Key exchange algorithm.**

**Algorithm :**

| **Alice** | **Bob** |
| --- | --- |
| Public Keys available = P, G | Public Keys available = P, G |
| Private Key Selected = a | Private Key Selected = b |
| Key generated =   x= Ga mod P | Key generated =  Y= Gb mod P |
| Exchange of generated keys takes place | |
| Key received = y | Key received = x |
| Algebraically, it can be shown that ka =kb | |
| Users now have a symmetric secret key to encrypt | |

#include<math.h>

#include<stdio.h>

// Power function to return value of a ^ b mod P

long long int power (long long int a, long long int b,long long int P)

{

    if (b == 1)

        return a;

    else

        return (((long long int) pow (a, b)) % P);

}

// Driver program

int main ()

{

    long long int P, G, x, a, y, b, ka, kb;

    // Both the persons will be agreed upon the

    // public keys G and P

    P = 23; // A prime number P is taken

    printf("The value of P : %lld\n", P);

    G = 9; // A primitive root for P, G is taken

    printf("The value of G : %lld\n\n", G);

    // Alice will choose the private key a

    a = 4; // a is the chosen private key

    printf("The private key a for Alice : %lld\n", a);

    x = power(G, a, P); // gets the generated key

    // Bob will choose the private key b

    b = 3; // b is the chosen private key

    printf("The private key b for Bob : %lld\n\n", b);

    y = power (G, b, P); // gets the generated key

    // Generating the secret key after the exchange

    // of keys

    ka = power (y, a, P); // Secret key for Alice

    kb = power (x, b, P); // Secret key for Bob

    printf("Secret key for the Alice is : %lld\n", ka);

    printf("Secret Key for the Bob is : %lld\n", kb);

    return 0;

}

**Input & Output:**

The value of P: 23  
The value of G: 9  
  
The private key a for Alice: 4  
The private key b for Bob: 3  
  
Secret key for the Alice is: 9  
Secret Key for the Bob is: 9