Lab 9

Name: Vaghani Smit Dhirubhai

Roll no: CE169

College Id: 19CEUEG022

Aim: Write a program to implement Elliptical Curve Cryptography.

- Key Generation
- Encryption
- Decryption

Program : Elliptical Curve Cryptography

Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef pair<int,int> point;

int p,a,b,n;

int multiplicative_inverse(int a,int b){
    int q,r,t,t1=0,t2=1,r1,r2;
    r1=b;
    r2=a;
    while(r2>0){
        q=r1/r2;
        r=r1-q*r2;
        r1=r2;
        r2=r;
```

```
        t=t1-q*t2;
        t1=t2;
        t2=t;
    }
    if(r1==1)
    {
        if(t1<0)
        t1+=b;
        return t1;
    }
    else//gcd!=1
    return 0;
}

int squareMultiply(int base, int exp, int mod)//base^exp(%
mod)
{
    int z=1;
    while(exp>0)
    {
        if(exp%2==1)
            z=(z*base)%mod;
        exp=exp/2;
        base=(base*base)%mod;
    }
    return z;
}
```

```
int mode(int a,int b)
{
    int ans=a%b;
    if(ans<0)
    ans+=b;
    return ans;
}
point operator+(point p1,point p2)
{
    point p3;
    int x1=p1.first,x2=p2.first,y1=p1.second,y2=p2.second;
    int lamda;
    if(x1!=x2 && y1 !=y2)
    {
        int t=x2-x1,t1=y2-y1;
        if(t<0)
        {
            t1=(-1)*t1;
            t=(-1)*t;
        }
        lamda=mode(t1*multiplicative_inverse(t,p),p);//((y2-
y1)/(x2-x1))mode p
    }
    else if(x1==x2 && y1==y2)
    {
        lamda=mode((3*x1*x1+a)*multiplicative_inverse(2*y1,p
),p);
    }
```

```
    p3.first=mode((lamda*lamda-x1-x2),p);// x3=(lamda^2 - x1
- x2)mode p
    p3.second=mode(lamda*(x1-p3.first)-y1,p);//
y3=(lamda(x1-x3) - y1)mode p
    return p3;
}
point operator*(point p1,int n)
{
    point ans;
    ans.first=p1.first;
    ans.second=p1.second;
    while(n>1)
    {
        ans=ans+p1;
        n--;
    }
    return ans;
}
vector<point> point_generate()
{
    int x=0;
    vector<point> points;
    while(x<p)
    {
        int temp=x*x*x+a*x+b;
        int w=temp%p;
        if(sqrt(w)*sqrt(w)==w)//perfect square
        {
```

```
            int t=sqrt(w);
            points.push_back(make_pair(x,t%p));
            if(t!=0)
            {
                t=-t;
                points.push_back(make_pair(x,t+p));
            }

        }
        else{
            int qr=squareMultiply(w,(p-1)/2,p);
            if(qr==1)
            {
                while(sqrt(w)*sqrt(w)!=w)
                {
                    w=w+p;
                }
                int t=sqrt(w);
                points.push_back(make_pair(x,t%p));
                if(t!=0)
                {
                    t=-t;
                    points.push_back(make_pair(x,t+p));
                }
            }
        }
        x=x+1;
    }
```

```
    return points;
}

int randomNumberBetweenRange(int n,int m)//n is included m
not included
{
    int num;
    srand(time(0));
    num=n+rand()%(m-n-1);
    return num;
}
bool isPointOnCurve(point p,vector<point> points)
{
    for(int i=0;i<points.size();i++)
    {
        if(p.first==points[i].first &&
p.second==points[i].second)
        return true;
    }
    return false;
}

vector<point> key_generate(int &d)
{
    vector<point> e;
    vector<point> points=point_generate();
    n=points.size();
    int index=randomNumberBetweenRange(0,n);
```

```
    point e1=points[index];
    // point e1=make_pair(1,4);
    d=4;
    point e2=e1*d;
    while(!isPointOnCurve(e2,points))
    {
        index=randomNumberBetweenRange(0,n);
        e1=points[index];
        e2=e1*d;
    }
    e.push_back(e1);
    e.push_back(e2);
    return e;
}

vector<point> encrypt(point e1,point e2,point m)
{
    vector<point> c;
    int r=1;
    cout<<"r="<<r<<endl;
    point c1=e1*r;
    point c2=m+e2*r;
    c.push_back(c1);
    c.push_back(c2);
    return c;
}
point decrypt(point c1,point c2,int d)
{
```

```
    point t=c1*d;
    point t1=make_pair(t.first,(-1)*t.second);//inverse of t
    return (c2+t1);
}

int main()
{
    cout<<"Enter a,b and p:\n";
    cin>>a>>b>>p;
    vector<point> e;
    int d;
    e=key_generate(d);
    point e1=e[0],e2=e[1];
    cout<<"e1 = ("<<e1.first<<","<<e1.second<<")"<<endl;
    cout<<"e2 = ("<<e2.first<<","<<e2.second<<")"<<endl;
    cout<<"d = "<<d;
    cout<<"\nenter plain text m:"<<endl;
    point m;
    cin>>m.first>>m.second;
    vector<point> c= encrypt(e1,e2,m);
    point c1=c[0],c2=c[1];
    cout<<"c1 = ("<<c1.first<<","<<c1.second<<")"<<endl;
    cout<<"c2 = ("<<c2.first<<","<<c2.second<<")"<<endl;
    point m1=decrypt(c1,c2,d);
    cout<<"m = ("<<m1.first<<","<<m1.second<<")"<<endl;
}
```

Test Case 1:

```
E:\NIS\Lab\lab 9>cd "e
yptography && "e:\NIS\
Enter a,b and p:
1 1 13
e1 = (4,11)
e2 = (11,11)
d = 4
enter plain text m:
12 5
r=1
c1 = (4,11)
c2 = (0,1)
m = (12,5)
```

Test Case 2:

```
Enter a,b and p:
2 3 67
e1 = (1,26)
e2 = (23,25)
d = 4
enter plain text m:
24 26
r=1
c1 = (1,26)
c2 = (21,44)
m = (24,26)
```