# Lab-9

**Name : Divan MunafSha Salimsha**
**Roll No : CE035**
**Subject : NIS**
**Student ID: 19CEUBG006**

**Aim:** **Write a program to implement Elliptical Curve Cryptography.**

- **Key Generation**
- **Encryption**
- **Decryption**

**Source Code:**

**Programming Language : C++**

**Ans:**

```
#include "functions.h"
#define Point pair<int, int>

struct Keys
{
    Point ep, e1, e2;
    int p;
    int d;
};
```

```
void printPoint(vector<Point> points)
{
    for (auto i : points)
        cout << "(" << i.first << " , " << i.second << ")" << endl;
}
```

```
Point addP(Point p1, Point p2, int a, int p)
{
    int x1 = p1.first, x2 = p2.first, y1 = p1.second, y2 = p2.second;
    int inverse = multiplicativeInverse(doModP(x2 - x1, p), p);
    int lamda = doModP(inverse * (y2 - y1), p) % p;
    int x3 = doModP((power(lamda, 2) - x1 - x2), p);
    int y3 = doModP(lamda * (x1 - x3) - y1, p);
    return make_pair(x3, y3);
}

Point multiplyP(int scalar, Point p1, int a, int p)
{
    if (scalar == 1)
    {
        return p1;
```

```
    }
    if (scalar == 2)
    {
        int x1 = p1.first, y1 = p1.second;
        int lamda = (((3 * power(x1, 2)) + a) * multiplicativeInverse(2 * y1, p)) %
p;
        int x3 = doModP((power(lamda, 2) - 2 * x1), p);
        int y3 = doModP((lamda * (x1 - x3) - y1), p);
        return make_pair(x3, y3);
    }
    if (scalar % 2 == 1)
    {
        Point p2 = multiplyP(scalar - 1, p1, a, p);
        return addP(p1, p2, a, p);
    }
    if (scalar % 2 == 0)
    {
        Point p2 = multiplyP(scalar / 2, p1, a, p);
        return multiplyP(2, p2, a, p);
    }
    return make_pair(-1, -1);
}

vector<Point> pointGeneration(int a, int b, int p)
{
    vector<Point> points;
    for (int x = 0; x < p; ++x)
    {
        int w = (power(x, 3) + (a * x) + b) % p;
        int rem = squareAndMultiply(w, ((p - 1) / 2), p);
        if (rem == 1)
        {
            while (sqrt(w) * sqrt(w) != w)
                w += p;
            points.push_back(make_pair(x, sqrt(w)));
            points.push_back((make_pair(x, doModP(-sqrt(w), p))));
        }
        else if (rem == 0)
            points.push_back(make_pair(x, 0));
    }
    return points;
}
```

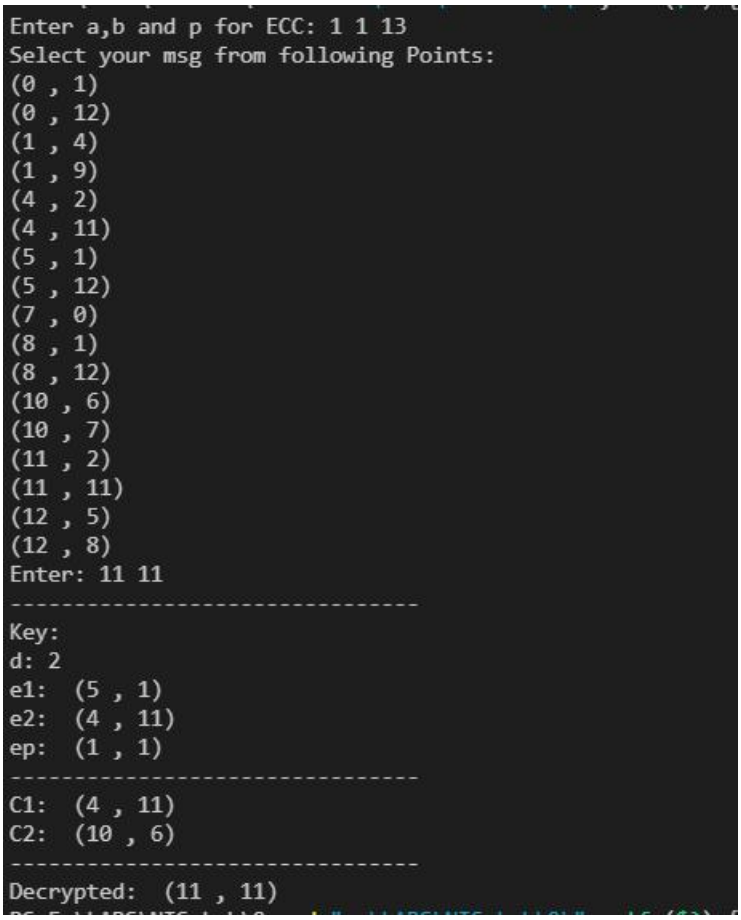
```
Keys keyGeneration(int a, int b, int p)
{
    Keys keys;
    vector<Point> points = pointGeneration(a, b, p);
    printPoint(points);
    keys.ep = make_pair(a, b);
    keys.d = 1 + (rand() % 10);
    keys.p = p;
    keys.e1 = points[1 + (rand() % points.size())];
    keys.e2 = multiplyP(keys.d, keys.e1, a, p);
    return keys;
```

```
}

void printP(Point p)
{
    cout << " (" << p.first << " , " << p.second << ")" << endl;
}

pair<Point, Point> encryption(Point M, Keys keys)
{
    int r = 1 + rand() % 3;
    Point c1 = multiplyP(r, keys.e1, keys.ep.first, keys.p);
    Point c2 = addP(M, multiplyP(r, keys.e2, keys.ep.first, keys.p), keys.ep.first,
keys.p);
    return make_pair(c1, c2);
}

Point decryption(pair<Point, Point> cipher, Keys keys)
{
    Point m, c1 = cipher.first, c2 = cipher.second;
    m = multiplyP(keys.d, c1, keys.ep.first, keys.p);
    m = make_pair(m.first, -m.second);
    Point M = addP(c2, m, keys.ep.first, keys.p);
    return M;
}

int main()
{
    pair<Point, Point> cipher;
    Point decrypted, msg;
    int a, b, p, m1, m2;
    cout << "Enter a,b and p for ECC: ";
    cin >> a >> b >> p;
    cout << "Select your msg from following Points: " << endl;
    Keys keys = keyGeneration(a, b, p);
    cout << "Enter: ";
    cin >> m1 >> m2;
    msg = make_pair(m1, m2);
    cout << "--------------------------------" << endl;
    cout << "Key: " << endl;
    cout << "d: " << keys.d << endl;
    cout << "e1: ";
    printP(keys.e1);
    cout << "e2: ";
    printP(keys.e2);
    cout << "ep: ";
    printP(keys.ep);
    cout << "--------------------------------" << endl;
    cipher = encryption(msg, keys);
    cout << "C1: ";
    printP(cipher.first);
    cout << "C2: ";
    printP(cipher.second);
    cout << "--------------------------------" << endl;
    Point d = decryption(cipher, keys);
```

```
    cout << "Decrypted: ";
    printP(d);
    return 0;
}
```

## Input & Output Screenshots:

**1)**




**2)**

```
Enter a,b and p for ECC: 2 3 67
Select your msg from following Points:
(1 , 26)
(1 , 41)
(2 , 22)
(2 , 45)
(3 , 6)
(3 , 61)
(5 , 2)
(5 , 65)
(7 , 5)
(7 , 62)
(8 , 14)
(8 , 53)
(11 , 4)
(11 , 63)
(13 , 22)
(13 , 45)
(17 , 27)
(17 , 40)
(21 , 23)
(21 , 44)
(23 , 25)
(23 , 42)
(24 , 26)
(24 , 41)
(25 , 0)
(26 , 12)
(26 , 55)
(28 , 13)
(28 , 54)
(29 , 14)
(29 , 53)
(30 , 14)
(30 , 53)
(35 , 1)
(35 , 1)
(35 , 66)
(42 , 26)
(42 , 41)
(43 , 0)
(50 , 9)
(50 , 58)
(51 , 30)
(51 , 37)
(52 , 22)
(52 , 45)
(55 , 23)
(55 , 44)
(57 , 16)
(57 , 51)
(58 , 23)
(58 , 44)
(63 , 20)
(63 , 47)
(64 , 29)
(64 , 38)
(66 , 0)
Enter: 55 45
--------------------------------
Key:
d: 2
e1:  (52 , 45)
e2:  (23 , 25)
ep:  (2 , 3)
--------------------------------
C1:  (23 , 25)
C2:  (14 , 5)
--------------------------------
Decrypted:  (55 , 45)
```