NIS

LAB - 7

ROll NO. : CE146

Name : Shingala Shubham P.

ID. NO. : 19CEUOS159.

* AIM : Write a Program to implement Elgamal Cryptosystem.
  → Function to create Primitive root for the give multiplicative group.
  → Key Generation. → Encryption
  → Decryption.

→ source code :

```
#include <bits/stdc++.h>
#define ll long long
#define v vector<ll>
#define loop(var, s, n) for(ll var=s; var<n; var++)
#define pb push_back
#define br cout<< endl;
using namespace std;

ll squareMultiply (ll base, ll exp, ll mod)
{
    ll z = 1;
    while(exp > 0)
    {
```

```
        if(exp % 2 == 1)
            z = (z * base) % mod;
        exp = exp/2;
        base = (base * base) % mod;
    }
    return z;
}

bool isPrime(ll n)
{
    if(n <= 1) return false;
    if(n <= 3) return true;
    if(n%2 == 0 || n%3 == 0) return false;
    for(ll i=5; i*i <= n; i+=6)
        if(n%i == 0 || n%(i+2) == 0)
            return false;
    return true;
}

ll randomNumberInRange(ll n, ll m)
{
    // n not included and m included.
    srand(time(0));
    ll random = n + rand() % (m-n-1);
    return random;
}
```

```
ll multiplicativeInverse(ll a, ll b)
{
    ll q, r, t, t1=0, t2=1, 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
        return -1;
}

V primitivRoot (V z_star; ll P)
{
    V roots;
    loop(i, 1, P-1)
    {
        loop(j, 1, P)
        {
```

```
            if( squareMultiply (z_star[i], j, P) == 1)
            {
                if (j < P-2)
                    break;
                else
                    roots.PB(z_star[i]);
            }
        }
    }
    return roots;
}

V encrypt (V z_star, ll e1, ll e2, ll m, ll P)
{
    V c(2);
    ll r = z_star[randomNumberInRange(0,P-1)];
    cout << "\n r = " << r; br
    c[0] = squareMultiply (e1, r, P);
    c[1] = (squareMultiply (e2, r, P) * m) % P;
    return c;
}

ll decrypt (V c, ll P, ll d)
{
    return (multiplicativeInverse (squareMultiply
                (c[0], d, P), P) * c[1]) % P;
}
```

(5)

```
int main()
{
    ll P;
    while (1)
    {
        cout << " Enter the large Prime
                number : " ;
        cin >> P;
        if(! isPrime(P))
            cout << P << " is not a Prime
                    number so, " ;
        else
            break;
    }

// Key Generation ----

V z-star;
loop(i, 1, P)   z-star.pb(i);

V roots = PrimitiveRoot(z_star, P);

ll d = z_star[RandomNumberIn
        Range(0, P-1)];
ll e1 = roots[RandomNumberIn
        Range(0, roots.size())];
ll e2 = squareMultiply(e1, d, P);
```

```
cout << " public key : int e1 = "
     << e1 << endl << "int e2 = " << e2
     << " int p = " << p <<
     " private key : int d = "<<d; br

ll m;
while (1)
{
    cout << " Enter the message
        less than prime number:";
    cin >> m;
    if (m >= P)
        cout << m << " is not less
         than " << P << " a prime
         number so, " ;
    else
        break;
}

v cipher = encrypt(z_star, e1, e2, m, P)
cout<<" Encryption numbers :"
    << cipher [0] << " and " << cipher[1]
cout << " Decryption message :"
    << decrypt (cipher, P, d); br
return 0;

}
```

(1)

→ Test - case - 1 :

Input : prime number : 89833
message : 43543

Key : public key :
e1 = 67481
e2 = 84701
p = 89833
private key :
d = 21243

Output : r = 7506
Encryption Number : 59421 and 12298
Decryption message : 43543

→ Test - case - 2 :

Input : prime number p = 1445
1445 is not prime so, Enter large
prime number : 4325
4325 is not prime so, Enter large
prime number : 7

Key : public key : e1 = 3, e2 = 2, p = 7
private key : d = 2

Enter message less than prime number : 145
because Elgamal Cryptosystem not work for

message greater than prime number

Enter message : 2

Output: r = 1

Encryption number : 3 and 4

Decryption message : 2

→ Test case - 3 :

Input : prime number : 6277

message : 5327

key :

Public key :

e1 = 2854, e2 = 693, p = 6277

Private key :

d = 5104

Output : r = 5131

Encryption number : 2535 and 1208

Decryption message : 5327