Lab – 5

Subject : NIS

Aim:To implement Elgamal algorithm for encryption and decryption.

Program: -

```
from random import randint
from MultiplyAndSquare import multiply_and_square as ms
from ExtendedEuclidian import multiplicative_inverse as mi

def primitive_roots_order(p):
    order=[None]*p
    for r in range(1,p):
        first_one=True
        for c in range(1,p):
            mas=ms(r,c,p)
            if((first_one) and (mas == 1)):
                order[r]=c
                first_one=False
    return order
def all_roots(order,p):
    proots=[]
    phi_p=phi(p)
    for i in range(1,len(order)):
        if(order[i] == phi_p):
            proots.append(i)
    return proots
def phi(a):
    return a-1
def generate_keys(p,proots):
    e1=proots[randint(0,len(proots)-1)]
    d=randint(1,p-2)
    e2=ms(e1,d,p)
    public=(e1,e2,p)
    private=d
    return public,private
def encrypt(m,e1,e2,p,proots):
```

```python
    r=proots[randint(0,len(proots)-1)]
    c1=ms(e1,r,p)
    c2=((e2**r)*m)%p
    return c1,c2
def decrypt(c1,c2,d,p):
    m= (c2*mi(c1,d))%p
    return m
if __name__ == "__main__":
    p=int(input("Enter large prime number:"))
    m=int(input("Enter Message:"))
    order = primitive_roots_order (p)
    proots = all_roots (order, p)
    public_key,private_key=generate_keys(p,proots)
    c1,c2=encrypt(m,public_key[0],public_key[1],public_key[2],proots)
    print("Encrypted messages:",c1,c2)
    dec_msg=decrypt(c1,c2,private_key,p)
    print("Decrypted message",dec_msg)
```

Output:-

```
PS D:\DDIT CE\Sem 6\NIS\Lab 5> python .\elgamal_cryptosystem.py
Enter large prime number:13
Enter Message:11
Encrypted messages: 12 11
Decrypted message 11
PS D:\DDIT CE\Sem 6\NIS\Lab 5>
```

Description:-

- Elgamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

⌧ Encryption:-

- In this algorithm first we have to generate big prime number.
- Then we have to find the all possible primitive roots of prime number which we had selected.

- Then after we have to generate public key and private key.
- Public key => (e1, e2, prime)
- Private key => (d, prime)
- For the 'e1' we have to chose random number from all primitive roots of prime.
- For e2 = e1^r mod prime, here r is random number from primitive roots.
- For the d chose random number from primitive roots
  Where, 1 <= d <= prime – 2.
- Now perform encryption operation:-
  C1 = (e1^r) mod prime
  C2 = (e2^r * M) mod prime
- We have two cipher text.

⊠ Decryption:-
- For the decryption of cipher text we need extended Euclidean algorithm and multiply and square algorithm.
- Plain_text = (c2 * ((c1^d)^-1)) mod prime.

⊠ Drawbacks:-
- Major drawback is for the finding all primitive roots of prime number is take very much time around n*n*log(n). here n is bigger prime number

- This algo is similar as RSA but here the encryption is strong.