Network Security PA5

Name: Kshitij Ramesh Tadkase
Roll: 18075029
Dept: CSE (B.Tech)

Github Link: https://github.com/KRT2305/NetSec-PA5

ElGamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.
This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group that is even if we know ga and gk, it is extremely difficult to compute gak.
In this cryptosystem, the original message M is masked by multiplying gak to it. To remove the mask, a clue is given in form of gk. Unless someone knows a, he will not be able to retrieve M. This is because finding discrete log in a cyclic group is difficult and simplifying knowing ga and gk is not good enough to compute gak.

El Gamal Encryption Algorithm has three parts:

- Key Generation:
  - The receiver chooses a very large number q and a cyclic group Fq.
  - From the cyclic group Fq, he choose any element g and an element a such that gcd(a, q) = 1
  - Then computes h = g^a
  - The receiver publishes F, h = ga, q, and g as his public key and retain a as private key

- Encryption:
  - Sender selects an element k from cyclic group F such that gcd(k, q) = 1.
  - Then computes p = g^k and s = h^k = g^(a*k).
  - Multiply s with M
  - Then send the receiver (p, M*s) = (g^k, M*s)

- Decryption:
  - Receiver calculates s′ = p^a = g^(a*k).
  - Receiver divides M*s by s′ to obtain M as s = s′.

Source Code:

```python
import random
from math import pow

a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a % b == 0:
        return b
    else:
        return gcd(b, a % b)

def gen_key(q):

    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)

    return key

def power(a, b, c):
    x = 1
    y = a

    while b > 0:
        if b % 2 != 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)

    return x % c
```

```python
def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q)
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])

    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])

    return en_msg, p

def decrypt(en_msg, p, key, q):

    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))

    return dr_msg
```

```python
def main():

    msg = 'encrypted'
    print("Original Message :", msg)

    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)

    key = gen_key(q)# Private key for receiver
    h = power(g, key, q)
    print("\n\ng used : ", g)
    print("g^a used : ", h)

    en_msg, p = encrypt(msg, q, h, g)
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    print("\n\nDecrypted Message :", dmsg);

main()
```

Results:

```
C:\Users\kshit>C:/Users/kshit/AppData/Local/Programs/Python/Python39/python.exe c:/Users/kshit/Desktop/pa5.py
Original Message : encrypted


g used :  14116918381753816243698580930083939631752780910377
g^a used :  23370649483470247447385034578965637571608066513893
g^k used :  10424748733478381640387947656234252608677141978025
g^ak used :  34383087572565207727171958543312475528220741273517


Decrypted Message : encrypted
```