Name - Aditya Patidar
Roll no. - 18075006
Dept. - CSE (B.Tech)

Network Security PA-5

Github Link -
https://github.com/Aditya-Patidar/Network-Security-Assignents-18075006-/tree/main/Practical%20Assignment%205

Objective Implement the El Gaamal algorithm in Python

Basic Theory

The Gamal encryption system is an example of asymmetric key encryption algorithm for public-key cryptography. It is based on Diffie–Hellman key exchange.The Digital Signature Algorithm (DSA) is a variant of the El Gamal signature scheme, which should not be confused with El Gamal encryption

Uses:- Hybrid cryptosystem uses this algorithm.

Algorithm: El Gamal Encryption Algorithm has three parts

- A key generator
- The encryption algorithm
- The decryption algorithm

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′.

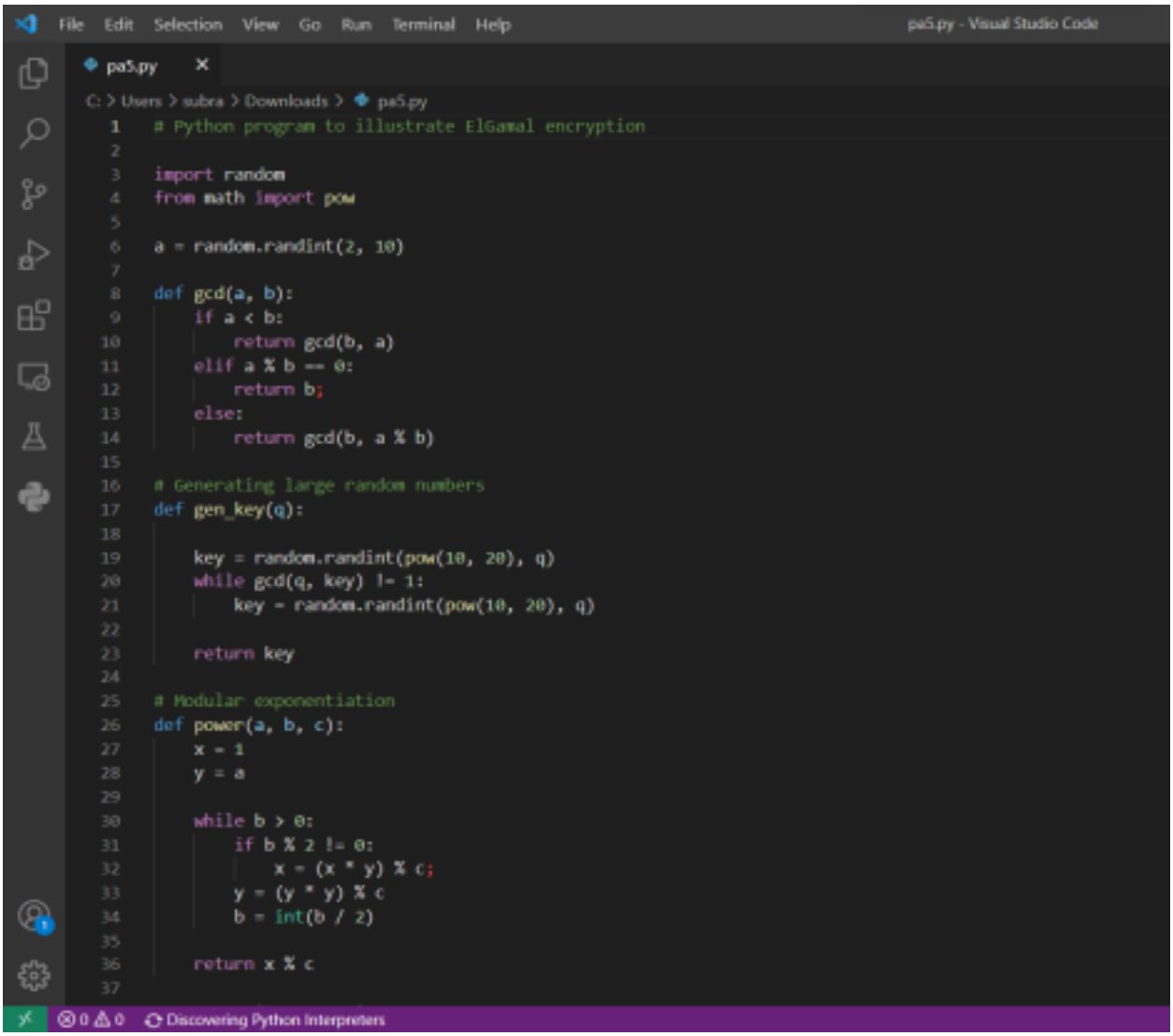
Source code:

```
# Python program to illustrate ElGamal encryption

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)

# Generating large random numbers
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

# Modular exponentiation
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

# Asymmetric encryption
def encrypt(msg, q, h, g):

        en_msg = []

        k = gen_key(q)# Private key for sender
        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

# Driver code
def main():

        msg = 'Implement the El Gaamal algorithm in Python'
        print("Message from senders side :", msg)

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

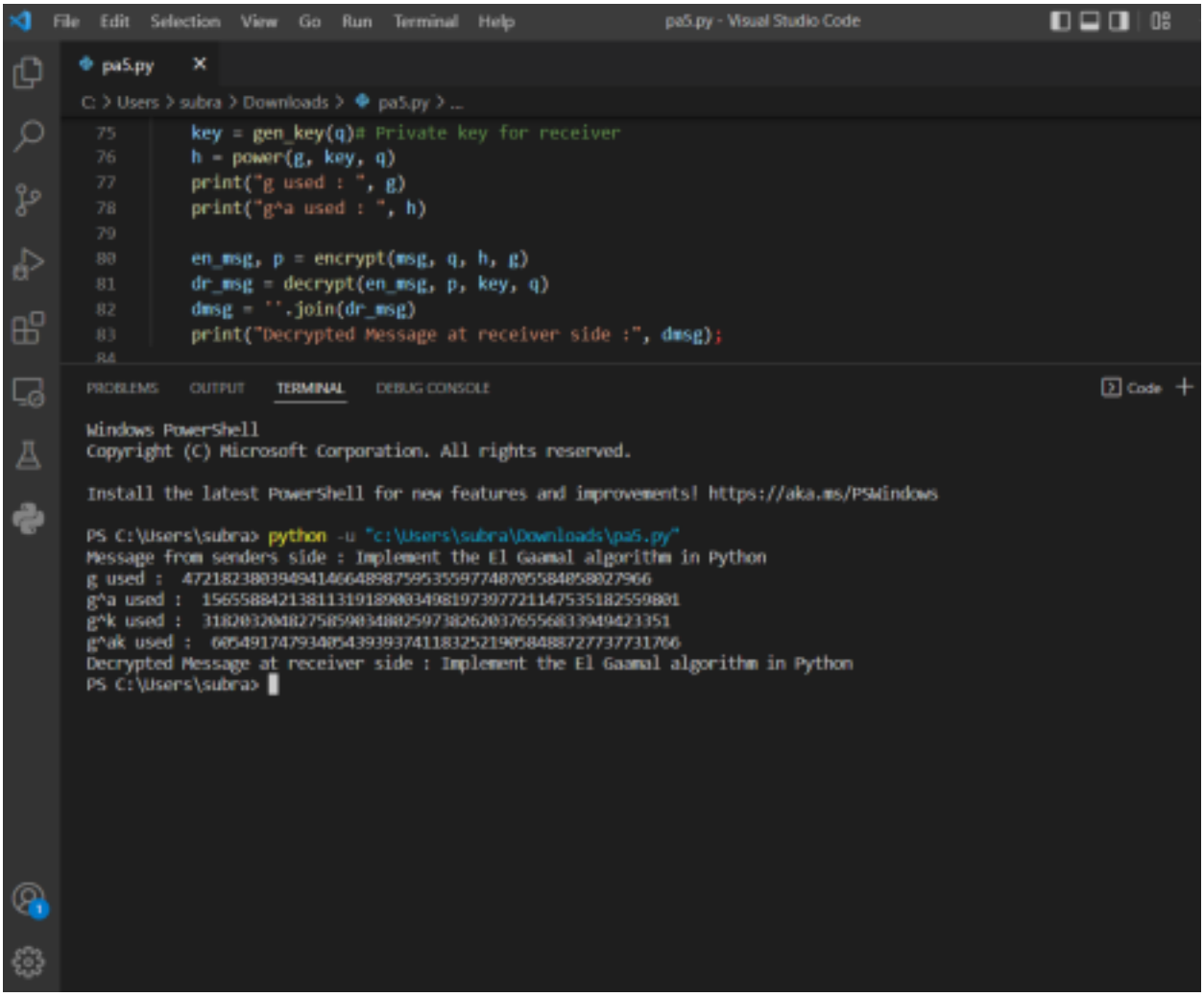
```
        g = random.randint(2, q)

        key = gen_key(q)# Private key for receiver
        h = power(g, key, q)
        print("g 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("Decrypted Message at receiver side :", dmsg);


if __name__ == '__main__':
        main()
```

Screenshots



```
    return key

# Modular exponentiation
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

# Asymmetric encryption
def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q)# Private key for sender
    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
```

```
    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

# Driver code
def main():

    msg = 'Implement the El Gaamal algorithm in Python'
    print("Message from senders side :", 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("g 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("Decrypted Message at receiver side :", dmsg);
```


File Edit Selection View Go Run Terminal Help
pa5.py - Visual Studio Code
pa5.py
C: > Users > subra > Downloads > pa5.py > ...
key = gen_key(q)# Private key for receiver
h = power(g, key, q)
print("g 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("Decrypted Message at receiver side :", dmsg);
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\subra> python -u "c:\Users\subra\Downloads\pa5.py"
Message from senders side : Implement the El Gaamal algorithm in Python
g used : 47218238039494146648987595355977487055840580279 66
g^a used : 15655884213811319189083498197397721147535182559881
g^k used : 31820320482758590348025973826203765568339419423351
g^ak used : 60549174793405439393741183252190584887277377317 66
Decrypted Message at receiver side : Implement the El Gaamal algorithm in Python
PS C:\Users\subra>