Practical Assignment 5

Divyansh Chaturvedi
18074005
CSE IDD

El Gamal encryption is named after its creator, Taher Elgamal, and is based on the DH Protocol. The procedure includes the following steps:

Assume that person Alice wishes to interact with person Bob; the first prerequisite would be to choose the two fixed, public parameters:

- p - 2048-bit prime number
- g - generator in the range $1 < g < p-1$

The recipient of the message, Bob, will pick a random number x in the range $0 <= b <= p-2$ and compute $X = g^x \bmod p$. Here, X is Bob's public key which Bob broadcasts to everyone, and x is Bob's private key which Bob will keep a secret.

1. Now that Alice is aware of Bob's public key, if Alice wishes to send a message msg to Bob, Alice will select a random number r from the range $0< r< p-2$. Alice will compute the ciphertext by computing its public key in order to encrypt the message msg as $R = g^r \bmod p$.
2. Next Alice will calculate the shared key $K = X^r \bmod p$.
3. The message msg will be encrypted by Alice by calculating the value of S = msg * K. The format of the ciphertext is (R,S), where R and S numbers fall in the range from 0 to p-1.
4. Then Alice will send to Bob the ciphertext, i.e (R,S).
5. Bob will then decrypt the received ciphertext by first calculating the value of $R^{-x} * S \bmod p$. Consequently Bob will find out the msg sent by Alice.

Each time A wishes to send a message to B, A selects a new random integer r and generates a new public key. B, on the other hand, will utilize the previously determined long-term public key.

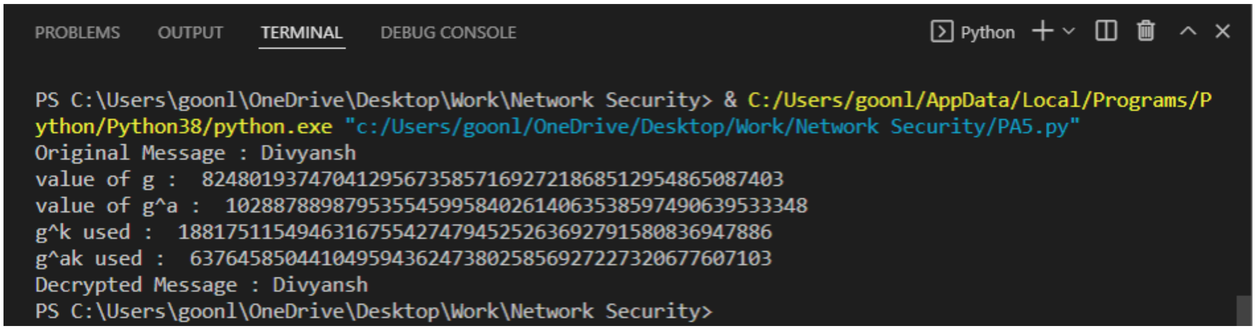
The El Gamal encryption's security is dependent on the approach of A using a fresh random number each time it has to send a message. As a result, the one-time pad is only used once. This one-time pad method is just as secure as XOR when used with modular exponentiation.

Code :

```
import random
from math import pow as pper

a = random.randint(2, 10)

def gcd(a, b):

    if a >= b:
        if a%b == 0:
            return b
        return gcd(b,a%b)

    return gcd(b,a)

def gen_key(q):

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


def power(a, b, c):
    if b == 0:
        return 1
    x = power(a,b//2,c)
    y = 1
    if b&1:
        y = a
    y = (x * x * y)%c
    return y
# Asymmetric encryption

def encrypt(msg, q, h, g):
    encrypted_msg = []
    k = gen_key(q)# Private key for sender
    s = power(h, k, q)
    p = power(g, k, q)
```

```python
 for i in range(0, len(msg)):
        encrypted_msg.append(msg[i])

    print("g^k used : ", p)

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

def decrypt(encrypted_msg, p, key, q):
    decrypted_msg = []
    h = power(p, key, q)
    for i in range(0, len(encrypted_msg)):

        decrypted_msg.append(chr(int(encrypted_msg[i]/h)))

    return decrypted_msg

# Driver code

def main():
    msg = 'Divyansh'
    print("Original Message :", msg)

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

    key = gen_key(q)# Private key for receiver
    h = power(g, key, q)
    print("value of g : ", g)
    print("value of g^a : ", h)

    encrypted_msg, p = encrypt(msg, q, h, g)

    decrypted_msg = decrypt(encrypted_msg, p, key, q)

    dmsg = ''.join(decrypted_msg)

    print("Decrypted Message :", dmsg);

if __name__ == '__main__':

    main()
```

Output :



Github Link - [Network Security PA 5]