Computer Networks

CSE-433

Practical Assignment - 5 (PA5)

Name - Aasav Badera

Roll No.- 18075001

Branch - Computer Science and Engineering

Year - 4 (B. Tech.)

Github Link for this Assignment- https://github.com/King-01/Network-Security/tree/main/PA5

El Gamal Encryption Algorithm -

It is an asymmetric algorithm, i.e. uses the asymmetric key for encryption of the message for inter-party communication.
It's based on the fact that it's difficult to find the discrete logarithm in a cyclic group, even if we know g^a and g^b, it's impossible to compute g^{ab}.

Algorithm -
Let's denote the sender of the message who will encrypt it using the receiver's key as A and the receiver of the message as B.

1. **Public and Private key generation at the receiver's end -**

- B will choose a very big number (Let's say M) and cyclic group(Let's say G).
- Now, B will also choose a number (Let's say base) and private key (Let's say R), such that - gcd(M, R) = 1.
- Now, h is calculated by the following equation -

$$h = (base^{\mathbf{R}})\%M$$

- base, M, G, h are published by B as the public key.

2. **Encryption at sender's end -**

- A chooses a private key (Let's say S) such that, gcd(S, M) = 1.
- Now, p is calculated by the following equation -

$$p = (base^{\mathbf{S}})\%M$$

- Now, comb is calculated by the following equation -

$$comb = (h^{\mathbf{S}})\%M$$

Thus, $comb = (base^{\mathbf{R * S}})\%M$.

- Now, A takes the original message (Let's say msg) and multiplies it with (comb) to get the encrypted message(Let's say enc_msg).
- Now, A transfers p and enc_msg to B(the receiver).

3. **Decryption at B's end -**

- B computes -

  $\text{comb1} = (p^{R})\%M \Rightarrow \text{comb1} = (\text{base}^{\wedge R*S})\%M.$

  Observation: comb1 = comb
- Thus, now we perform the division of enc_msg(msg*comb) with comb1(comb) to get the original message(msg).

Code WorkFlow -

- The code takes a message that needs to be communicated as input from the user.
- Function exponentiate_mod(num, power, mod) does the modular exponentiation
- Function generate_key(modval) generates a random key such that gcd(key,modval)=1.

Code implementation -

```
# Python program to illustrate ElGamal encryption

import random
from math import pow, gcd

a = random.randint(2, 10)
l, r = pow(10, 28), pow(10, 60)
oq = {}
# Generating large random numbers
def generate_key(M):
    while True:
        random_trial = random.randint(l, M)
        if gcd(random_trial, M) == 1:
            return random_trial
```

```python
# Modular exponentiation
def exponentiate_mod(a, p, M):
    x, y = 1, a
    while p > 0:
        if p & 1:
            x = (x * y) % M
        y = (y * y) % M
        p = p // 2

    return x % M


# Encrypts the message rec_text from receiver
def encrypt_message(rec_text, M, h, g):

    en_rec_text, k = [], generate_key(M)

    s, p = exponentiate_mod(h, k, M), exponentiate_mod(g, k, M)

    for i in range(0, len(rec_text)):
        en_rec_text.append(rec_text[i])
    for i in range(0, len(en_rec_text)):
        en_rec_text[i] = s * ord(en_rec_text[i])
    global oq
    oq["s"] = s
    return en_rec_text, p


# Decrypts the encrypted message en_rec_text from receiver
def decrypt_message(en_rec_text, p, key, M):

    dr_rec_text, h = [], exponentiate_mod(p, key, M)

    for i in range(0, len(en_rec_text)):
        dr_rec_text.append(chr(int(en_rec_text[i] / h)))

    return dr_rec_text


if __name__ == "__main__":
    rec_text = input("Enter the message to send - ")

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

```python
    g, key = random.randint(2, M), generate_key(M)
    # key - private key for receiver
    h = exponentiate_mod(g, key, M)
    # Encrypt the message from the sender
    en_rec_text, p = encrypt_message(rec_text, M, h, g)
    # Decrypt the encrypted message, once receiver receives it
    decrypted_rec_text = decrypt_message(en_rec_text, p, key, M)
    # Construct the decrypted text from array of strings
    decrypted_rec_text = "".join(decrypted_rec_text)

    print(
        "Received entry(Original message) - {}\ng value - {}\nh value - {}\np
value - {}\ns value - {}\nDecrypted Message - {}".format(
            rec_text, g, h, p, oq["s"], decrypted_rec_text
        )
    )
```

Output -

```
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.

D:\Downloads\Assignments\Practical Assignment-4 (PA4)>C:/Users/Asus/anaconda3/Scripts/activate

(base) D:\Downloads\Assignments\Practical Assignment-4 (PA4)>conda activate base

(base) D:\Downloads\Assignments\Practical Assignment-4 (PA4)>C:/Users/Asus/anaconda3/python.exe "d:/Downloads/Ass
ignments/Practical Assignment-4 (PA4)/elgamal.py"
Enter the message to send - You should run 2 hours a day!
Received entry(Original message) - You should run 2 hours a day!
g value - 1862261523751829670619694740106432050640102885430 0
h value - 61849831808482533830546547349915022895111273694431
p value - 24427270032244135347379703630318473842397360643889
s value - 56368162707566572376999225284725701837405565932 34
Decrypted Message - You should run 2 hours a day!

(base) D:\Downloads\Assignments\Practical Assignment-4 (PA4)>
```