Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 17-09-2022	Subject: Cryptology

Aim

To implement ElGamel Algorithm.

Theory

In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique [ELGA84, ELGA85]. The Elgamal[2] cryptosystem is used in some form in a number of standards including the digital signature standard (DSS), which is covered in Chapter 13, and the S/MIME e-mail standard (Chapter 19).

As with Diffie-Hellman, the global elements of Elgamal are a prime number q and , which is a primitive root of q. User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 \; 6 \; X_A \; 6 \; q - 1$.
2. Compute $Y^A = a^{X_A} \bmod q$.
3. A's private key is X_A and A's public key is $\{q, a, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \ldots M \ldots q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.
2. Choose a random integer k such that $1 \ldots k \ldots q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = a^k \bmod q; \; C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$.
2. Compute $M = (C_2 K^{-1}) \bmod q$.

These steps are summarized in Figure 10.3. It corresponds to Figure 9.1a: Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.

Let us demonstrate why the Elgamal scheme works. First, we show how K is recovered by the decryption process:

$K = (Y_A)^k \bmod q$	K is defined during the encryption process
$K = (a^{X_A} \bmod q)^k \bmod q$	substitute using $Y_A = a^{X_A} \bmod q$ K
$= a^{kX_A} \bmod q$	by the rules of modular arithmetic
$K = (C_1)^{X_A} \bmod q$	substitute using $C_1 = a^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$(C_2K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

Global Public Elements	
q	prime number
a	a 6 q and a a primitive root of q

Key Generation by Alice	
Select private X_A	X_A 6 $q - 1$
Calculate Y_A	$Y_A = \mathrm{a}^{X_A} \bmod q$
Public key	$\{q, \mathrm{a}, Y_A\}X_A$
Private key	

Encryption by Bob with Alice's Public Key	
Plaintext:	M 6 q
Select random integer k	k 6 q
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \mathrm{a}^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2K^{-1}) \bmod q$

1. Bob generates a random integer k.
2. Bob generates a one-time key K using Alice's public-key components Y_A, q, and k.
3. Bob encrypts k using the public-key component a, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C_1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from C_2.

Thus, K functions as a one-time key, used to encrypt and decrypt the message.

For example, let us start with the prime field GF(19); that is, q 19. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 8.3. We choose $a = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = a^{X_A} \bmod q$ $a^5 \bmod 19$ 3 (see Table 8.3).
3. Alice's private key is 5 and Alice's public key is $\{q, a, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then:

1. Bob chooses k 6.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19$ 729 mod 19 7.
3. So

   $C_1 = a^k \bmod q = a^6 \bmod 19 = 11$

   $C_2 = KM \bmod q = 7 * 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2K^{-1}) \bmod q = 5 * 11 \bmod 19 = 55 \bmod 19 = 17$.

If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block M_1 of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = a^k \bmod q;\ C_{2,1} = KM_1 \bmod q$$
$$C_{1,2} = a^k \bmod q;\ C_{2,2} = KM_2 \bmod q$$

Then,

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

If M_1 is known, then M_2 is easily computed as

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

The security of Elgamal is based on the difficulty of computing discrete logarithms. To recover A's private key, an adversary would have to compute $X_A = \text{dlog}_{a,q}(Y_A)$. Alternatively, to recover the one-time key K, an adversary would have to determine the random number k, and this would require computing the discrete logarithm $k = \text{dlog}_{a,q}(C_1)$. [STIN06] points out that these calculations
are regarded as infeasible if p is at least 300 decimal digits and $q - 1$ has at least one "large" prime factor.

Code

```
print("Key Generation Process")
p = int(input("Enter a prime number : "))
d = int(input("Enter a decryption key : "))
e1 = int(input("Enter the 2nd part of Encryption Key : "))
e2 = pow(e1,d) % p
print("The 3rd Part of Encryption Key is : ",e2)
print("The Public Key is : ",[e1,e2,p])
print("\n--------------------------------------------")
print("\nEncrption Process")
r = int(input("Enter a random integer : "))
c1 = pow(e1,r)%p
print("Computer Cipher text 1 is : ",c1)
pt = int(input("Enter the lenght of Plain Text : "))
c2 = pt * pow(e2,r)%p
print("Computer Cipher text 2 is : ",c2)
print("The Cipher Text is : ",[c1,c2])
print("\n--------------------------------------------")
print("\nDecryption Process")
x = pow(c1,d)
i = 1
while True:
    if(i*x % p == 1):
        D = i
        break
    i += 1

PT = (c2*D)%p
print("The Plain Text Length is : ",PT)
```

Output

```
PS E:\College-Codes\Fourth Year\SEM VII> & C:/Users/varun/AppData/Local/Programs/Python/Python310/python.exe "e:/College-Cod
es/Fourth Year/SEM VII/CT/elagamel_Practical_7.py"
Key Generation Process
Enter a prime number : 11
Enter a decryption key : 3
Enter the 2nd part of Encryption Key : 2
The 3rd Part of Encryption Key is :  8
The Public Key is :  [2, 8, 11]
Computer Cipher text 1 is :  5
Enter the lenght of Plain Text : 7
Computer Cipher text 2 is :  6
The Cipher Text is :  [5, 6]

--------------------------------------------

Decryption Process
The Plain Text Length is :  7
```

Conclusion

Hence, we were able to perform ElGamel Algorithm.