EXP 8

NAME: DIVYESH KHUNT SAPID:60009210116 BATCH:D12

CHAPTER 10. OTHER PUBLIC KEY CRYPTOSYSTEMS

10.1. Diffie-Hellman Key Exchange

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages.

if a is “a” primitive root of the prime number p, then the numbers

$$a \bmod p, a^2 \bmod p, \ldots, a^{p1} \bmod p$$

are distinct and consist of the integers from 1 through p -1 in some permutation.

Example:

3 is primitive root of 7.

The number 3 is a primitive root modulo 7, because

$$3^1 = 3 = 3^0 \times 3 \equiv 3 \equiv 3\ (m\ \mathrm{o}\ d)7$$

$$3^2 = 9 = 3^2 \times 3 \equiv 9 \equiv 2\ (m\ \mathrm{o}\ d\ \ 7)$$

$$3^3 = 27 = 3^3 \times 3 \equiv 6 \equiv 6\ (m\ \mathrm{o}\ d\ \ 7)$$

$$3^4 = 81 = 3^4 \times 3 \equiv 18 \equiv 4\ (m\ \mathrm{o}\ d\ \ 7)$$

$$3^5 = 243 = 3^5 \times 3 \equiv 12 \equiv 5\ (m\ \mathrm{o}\ d\ \ 7)$$

$$3^6 = 729 = 3^6 \times 3 \equiv 15 \equiv 1\ (m\ \mathrm{o}\ d\ \ 7)$$

Here we see that the period of 3^k modulo 7 is 6. The remainders in the period, which are 3, 2, 6, 4, 5, 1, form a rearrangement of all nonzero remainders modulo 7, implying that 3 is indeed a primitive root modulo 7.

The Algorithm:

There are two publicly known numbers: a prime number q and an integer that is a primitive root of q.

Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{XA}$ mod q. Similarly, user B independently selects a random integer $X_A < q$ and computes $Y_B = \alpha^{XB}$ mod q.

Each side keeps the X value private and makes the Y value available publicly to the other side.

User A computes the key as $K = (Y_B)^{X_A}$mod q and user B computes the key as $K = (Y_A)^{XB}$ mod q. These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{XA} \bmod q \\
&= (\alpha^{XB} \bmod q)^{XA} \bmod q \\
&= (\alpha^{XB})^{XA} \bmod q \qquad \text{by the rules of modular arithmetic} \\
&= (\alpha^{XB\,XA} \bmod q \\
&= (\alpha^{XA})^{XB} \bmod q \\
&= (\alpha^{XA} \bmod q) \\
&= (\alpha^{XA} \bmod q)^{XB} \bmod q \\
&= (Y_A)^{XB} \bmod q
\end{aligned}
$$

Example:

Prime number q = 353 and a primitive root of 353, in this case $\alpha = 3$.

A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes Y_A = 397 mod 353 = 40.

B computes Y_B = 3233 mod 353 = 248.

After they exchange public keys, each can compute the common secret key:

A computes K $= (Y_B)^{XA}$ mod 353 = 24897 mod 353 =160.

B computes K $= (Y_A)^{XB}$ mod 353 = 40233 mod 353.

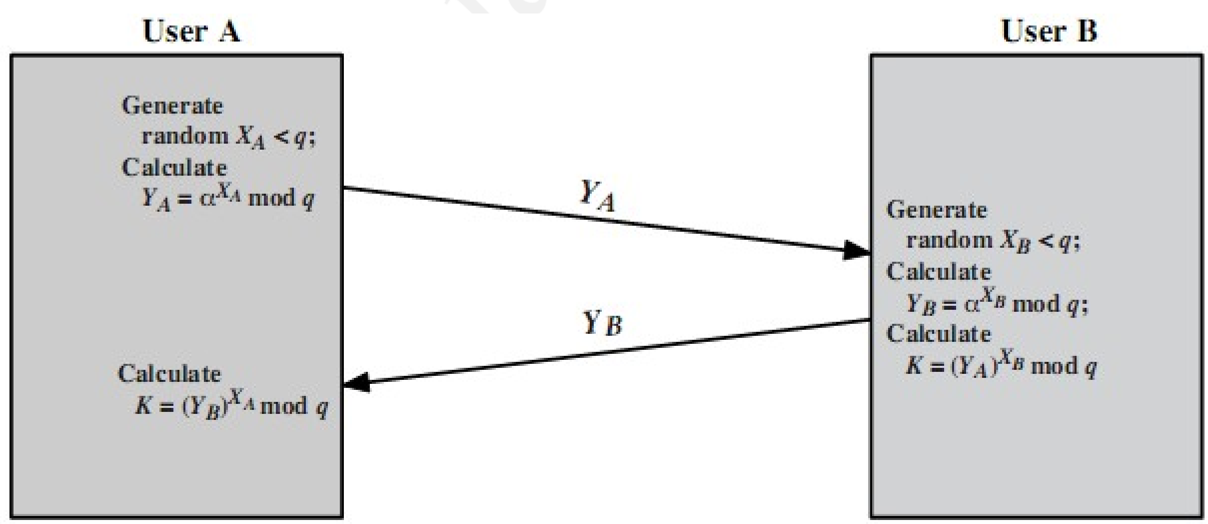
Key Exchange Protocols:

Figure 10.8 shows a simple protocol that makes use of the Diffie-Hellman calculation.

Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B.

User B responds by generating a private value X_B calculating Y_B, and sending Y_B to user A. Both users can now calculate the key.

The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.



Man-in-the-Middle Attack:

The protocol depicted in Figure 10.2 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows.

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates
   $$K2 = (Y_A)^{XD2} \bmod q$$
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{XB} \bmod q$
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits to Alice. Darth calculates $K2 = (Y_B)^{XD1} \bmod q$
7. Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{XA} \bmod q$

.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2 .All future communication between Bob and Alice is compromised in the following way.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates;

10.2 ELGAMAL CRYPTOGRAPHIC SYSTEM

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

1. Generate a random integer X_A , such that $1 < X_A < Q-1$
2. Compute $Y_A = \alpha^{XA} \bmod q$
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$

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 $0 \leq M \leq q - 1$. 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 $1 \leq k \leq q - 1$
3. Compute a one-time key $K = (Y_A)^k \bmod q$

Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q$$

$$C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

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

User A generates a public/private key pair; user B encrypts using A's public key; and user A 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
$K = (\alpha^{X_A} \bmod q)^k \bmod q$	Substitute using $Y_A = \alpha^{X_A} \bmod q$
$K = (\alpha^{X_A k} \bmod q) \bmod q$	By the rules of modular arithmetic
$K = (C_1)^{X_A} \bmod q$	Substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$M = (C_2 K^{-1}) \bmod q$$

$$= KMK^{-1} \bmod q$$

$$= M \bmod q = M$$

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
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	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Example:

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, and 15}, as shown in Table 8.3.We choose $\alpha = 10$.

Encryption:

User A generates a key pair as follows:

1. User A chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = 10^5$ mod 19= 3 (100000 mod 19=3).
3. A's private key is 5;Alice's public key is $\{q, \alpha, Y_A\} = \{19,10,3\}$

Suppose User B wants to send the message with the value M =17.Then,

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

   So

$$C_1 = \alpha^k \bmod q = 10^6 \bmod 19 = 11$$

$$C_2 = KM \bmod q = 7 * 17 \bmod 19 = 119 \bmod 19 = 5.$$

3. User B sends a Ciphertext as (11,5).

Decryption:

1. User A calculates

$$K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 16101 \bmod 19 = 7.$$

2. Then K^{-1} in $GF(19)$ is $7^{-1} \bmod 19 = 11$.
3. Finally

$$M = (C_2 K^{-1}) \bmod q = (5 * 11) \bmod 19 = 55 \bmod 19 = 17.$$

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 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

$$C_{1,1} = \alpha^k \bmod q; \quad C_{2,1} = K M_1 \bmod q$$

$$C_{1,2} = \alpha^k \bmod q; \quad C_{2,2} = K M_2 \bmod q$$

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

If M_1 is known M_2 can be calculated as,

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

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 $X_Æ = dlog_{a,q}(Y_Æ)$.(discrete Lograthms)

```python
print("ENTER PRIME NUMBER\n")
p = int(input("Enter value of p:"))
g = int(input("Enter value of q:"))
a = int(input("Enter value of a:"))
b = int(input("Enter value of b:"))

X,y  = g**a % p,g**b % p
print("X=", X)
print("Y =", y)

k1,k2 = y**a % p,X**b % p
print("k1 =", k1)
print("k2 =", k2)
```

```
ENTER PRIME NUMBER

Enter value of p:23
Enter value of q:9
Enter value of a:4
Enter value of b:3
X= 6
Y = 16
k1 = 9
k2 = 9
```

```
def prime_checker(p):
  if p < 1:
    return -1
  elif p > 1:
    if p == 2:
      return 1
    for i in range(2, p):
      if p % i == 0:
        return -1
      return 1

def primitive_check(g, p, L):
  for i in range(1, p):
    L.append(pow(g, i) % p)
  for i in range(1, p):
    if L.count(i) > 1:
      L.clear()
      return -1
    return 1

l = []
while 1:
  P = int(input("Enter P : "))
  if prime_checker(P) == -1:
    print("Number Is Not Prime")
    continue
  break

while 1:
  G = int(input(f"Enter The Primitive Root Of {P} : "))
  if primitive_check(G, P, l) == -1:
    print(f"Number Is Not A Primitive Root Of {P}")
    continue
  break
```

```
Enter P : 23
Enter The Primitive Root Of 23 : 9
Number Is Not A Primitive Root Of 23
Enter The Primitive Root Of 23 : 5
```

```
[6] l
```

```
[5, 2, 10, 4, 20, 8, 17, 16, 11, 9, 22, 18, 21, 13, 19, 3, 15, 6, 7, 12, 14, 1]
```