ST. FRANCIS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY
SECURITY LAB

Experiment – 6: Implementation and Simulation of Diffie-Hellman Key Exchange Algorithm

Aim: To implement Diffie Hellman (DH) Key Exchange algorithm.

Objective: After performing the experiment, the students will be able to understand the Diffie-Hellman Key exchange algorithm.

Lab objective mapped: L502.1: Students should be able to apply the knowledge of symmetric cryptography to implement simple ciphers.
L502.2: Students should be able to analyse and implement public key algorithms.

Prerequisite: Basic knowledge of symmetric key cryptography

Requirements: C/C++/JAVA/PYTHON

Pre-Experiment Theory:

Diffie Hellman (DH) key exchange is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Algorithm:
Alice and Bob are two users who wish to establish secure communication using Diffie Hellman algorithm. We can assume that Alice and Bob know nothing about each other but are in communication.

Global Public Elements

$q \qquad \textit{prime number}$

$\alpha \qquad \alpha \le q$

User A key Generation

select private $X_A \qquad X_A \le q$

calculate public $Y_A \qquad Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

select private $X_B \qquad X_B \le q$

calculate public $Y_B \qquad Y_B = \alpha^{X_B} \bmod q$

Calculatio n of Secret key by User A

$$K = (Y_B)^{X_A} \bmod q$$

Calculatio n of secret key by user B

$$K = (Y_A)^{X_B} \bmod q$$

As shown above,

1. Alice and Bob agree on two large positive integers, q and α, where p is a prime number and α is a primitive root mod q.

2. Alice randomly chooses another large positive integer, X_A, which is smaller than q. X_A will serve as Alice's private key.
3. Bob similarly chooses his own private key, X_B.
4. Alice computes her public key, Y_A, using the formula $Y_A = \alpha^{X_A} \, mod \, q$
5. Bob similarly computes his public key, Y_B, using the formula $Y_B = \alpha^{X_B} \, mod \, q$
6. Alice and Bob exchange public keys over the insecure channel.
7. Alice computes the shared secret key, K, using the formula $K = Y_B^{X_A} \, mod \, q$
8. Bob computes the same shared secret key, K, using the formula $K = Y_A^{X_B} \, mod \, q$
9. Alice and Bob communicate using the symmetric algorithm of their choice and the shared secret key, K, which was never transmitted over the insecure channel.

Procedure:

1. Access the Cryptography virtual lab link http://cse29-iiith.vlabs.ac.in/
2. Goto 'List of Experiments' – 'Diffie-Hellman Key Establishment' - 'Simulation' tab.
3. Understand the simulation process of Diffie Hellman Key exchange. Follow the key exchange process by generating prime and generator numbers. Take screenshot.
4. Write a program in C/C++/Java or Python for Diffie Hellman Key exchange.
    a. For Key generation, ask user to enter the value of prime number q & α and public keys of A and B, X_A & X_B. (Note that values of q, α, X_A & X_B cannot be random, they should satisfy criteria as per DH algorithm)
    b. Program should calculate the private keys for both Alice and Bob as per DH algorithm.
    c. Provide a set of public (e, n) and private key (d, n) of Alice and Bob as the output to the user.
5. Test the output of program for following exercise:
    a. Alice and Bob decide to use Diffie Hellman key exchange with $q = 23, \propto = 7, X_A = 3$ & $X_B = 6$. Find their public and private keys and the shared key.
    b. For $q = 7, \propto = 3, X_A = 2$ & $X_B = 5$ Validate public keys (Ans: 2 &5), private keys (Ans: 2 & 5) and the shared key (Ans: K=4).
    c. For $q = 17, \propto = 5, X_A = 4$ & $X_B = 6$ Validate public keys (Ans: 13 & 2), private keys (Ans: 4 & 6) and the shared key (Ans: K=16).

Output:

1. Attach the complete code performing key generation of both the parties.
2. Attach the program output for key generation (display public key & private key for Both A & B) for the inputs given in all three exercises above.
3. Attach the Screenshot taken for key generation using Cryptography virtual lab simulation.

Post Experimental Exercise-

1. Solve any two exercises mentioned in the procedure on the journal sheet. [Theoretical result and attached code's output should match].
2. Discuss five practical applications of Diffie Hellman key exchange algorithm.

Conclusion:

The Diffie-Hellman key exchange algorithm is a fundamental cryptographic technique that enables two parties to securely establish a shared secret key over an insecure communication channel. This shared key can then be used for various encryption and authentication purposes.

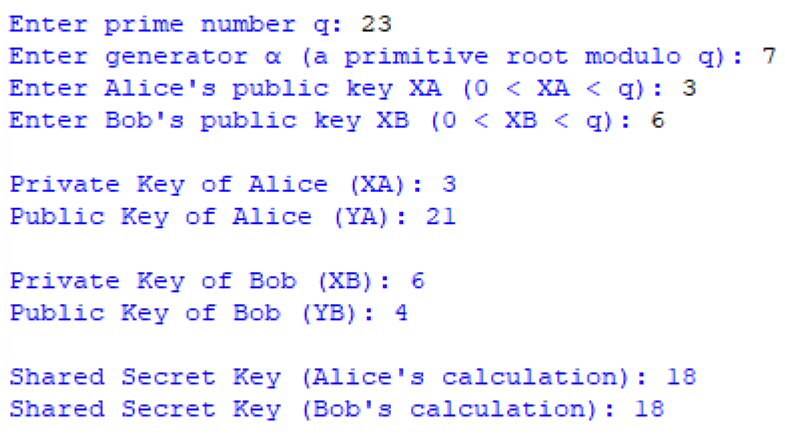
References: Virtual Cryptography Lab link: http://cse29-iiith.vlabs.ac.in/

<u>Code:</u>

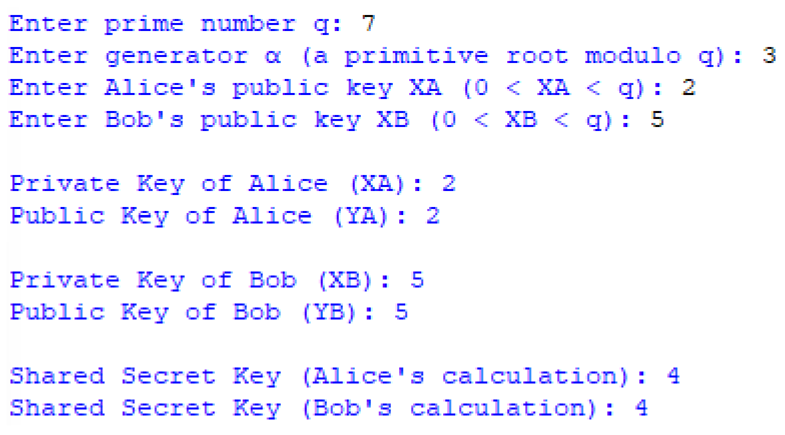
```
import random

# Function to calculate the modular exponentiation (x^y mod p)

def mod_exp(x, y, p):

    result = 1

    x = x % p

    while y > 0:

        if y % 2 == 1:

            result = (result * x) % p

        y = y // 2

        x = (x * x) % p

    return result

# Function to calculate the modular multiplicative inverse (a^(-1) mod m)

def mod_inverse(a, m):

    for x in range(1, m):

        if (a * x) % m == 1:

            return x

    return None

# Input from user

q = int(input("Enter prime number q: "))

alpha = int(input("Enter generator α (a primitive root modulo q): "))

XA = int(input("Enter Alice's public key XA (0 < XA < q): "))

XB = int(input("Enter Bob's public key XB (0 < XB < q): "))

# Calculate private keys

YA = mod_exp(alpha, XA, q)

YB = mod_exp(alpha, XB, q)
```

```
# Compute the shared secret key

shared_secret_A = mod_exp(YB, XA, q)

shared_secret_B = mod_exp(YA, XB, q)

# Output keys

print("\nPrivate Key of Alice (XA):", XA)

print("Public Key of Alice (YA):", YA)

print("\nPrivate Key of Bob (XB):", XB)

print("Public Key of Bob (YB):", YB)

print("\nShared Secret Key (Alice's calculation):", shared_secret_A)

print("Shared Secret Key (Bob's calculation):", shared_secret_B)
```

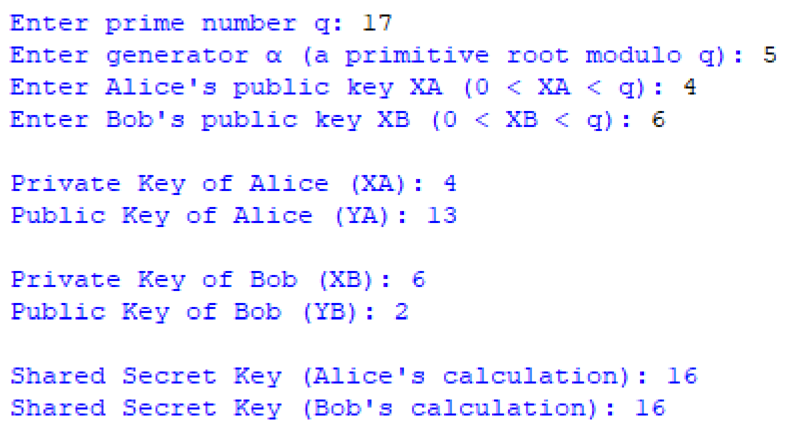
Output:

1.


```
Enter prime number q: 23
Enter generator α (a primitive root modulo q): 7
Enter Alice's public key XA (0 < XA < q): 3
Enter Bob's public key XB (0 < XB < q): 6

Private Key of Alice (XA): 3
Public Key of Alice (YA): 21

Private Key of Bob (XB): 6
Public Key of Bob (YB): 4

Shared Secret Key (Alice's calculation): 18
Shared Secret Key (Bob's calculation): 18
```

2.


```
Enter prime number q: 7
Enter generator α (a primitive root modulo q): 3
Enter Alice's public key XA (0 < XA < q): 2
Enter Bob's public key XB (0 < XB < q): 5

Private Key of Alice (XA): 2
Public Key of Alice (YA): 2

Private Key of Bob (XB): 5
Public Key of Bob (YB): 5

Shared Secret Key (Alice's calculation): 4
Shared Secret Key (Bob's calculation): 4
```

3.


```
Enter prime number q: 17
Enter generator α (a primitive root modulo q): 5
Enter Alice's public key XA (0 < XA < q): 4
Enter Bob's public key XB (0 < XB < q): 6

Private Key of Alice (XA): 4
Public Key of Alice (YA): 13

Private Key of Bob (XB): 6
Public Key of Bob (YB): 2

Shared Secret Key (Alice's calculation): 16
Shared Secret Key (Bob's calculation): 16
```