**ST. FRANCIS INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SECURITY LAB**

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

**Aim:** Write a program 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:** L502.2: Students should be able to analyse and implement public key algorithms.

**Prerequisite:** Basic knowledge of asymmetric key cryptography.

**Requirements:** PYTHON

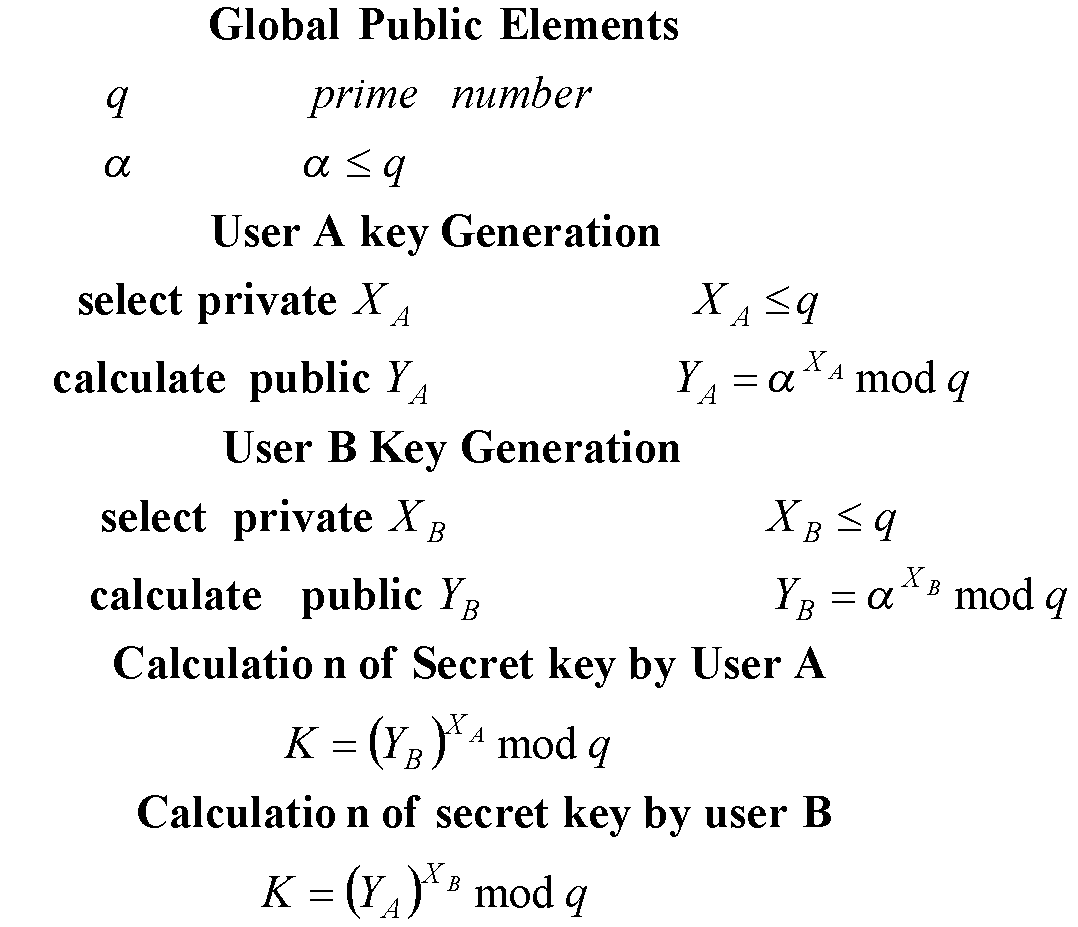
**Pre-Experiment Theory:**

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. The efficiency of this algorithm is based on computation of discrete logarithm problem for large prime values.

**Algorithm:**

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

**Steps:**



As shown above,

1. Alice and Bob agree on two large positive integers, q and α, where q is a prime number and α is a primitive root mod q.
2. Alice randomly chooses another large positive integer, , which is smaller than q. will serve as Alice's private key.
3. Bob similarly chooses his own private key, .
4. Alice computes her public key, , using the formula
5. Bob similarly computes his public key, , using the formula
6. Alice and Bob exchange public keys over the insecure channel.
7. Alice computes the shared secret key, , using the formula
8. Bob computes the same shared secret key, , using the formula
9. Alice and Bob communicate using the symmetric algorithm of their choice and the shared secret key, , which was never transmitted over the insecure channel.

**Procedure:**

1. Access the Cryptography virtual lab link [http://cse29-iiith.vlabs.ac.in/](http://cse29-iiith.vlabs.ac.in/%20)
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 inPython for Diffie Hellman Key exchange.
   1. For Key generation, ask user to enter the value of prime number q & α and public keys of A and B, &. (Note that values of q, α, & cannot be random, they should satisfy criteria as per DH algorithm)
   2. Program should calculate the private keys for both Alice and Bob as per DH algorithm.
   3. 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:
   1. Alice and Bob decide to use Diffie Hellman key exchange with. Find their public and private keys and the shared key.
   2. For Validate public keys (Ans: 2 &5), private keys (Ans: 2 & 5) and the shared key (Ans: K=4).
   3. For 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 all three 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/

Code:

def mod\_exp(base, exponent, modulus):

"""Function to perform modular exponentiation."""

result = 1

base = base % modulus

while exponent > 0:

if exponent % 2 == 1:

result = (result \* base) % modulus

exponent = exponent >> 1

base = (base \* base) % modulus

return result

def diffie\_hellman\_key\_exchange(q, alpha, XA, XB):

"""Function to perform Diffie-Hellman Key Exchange."""

# Calculate public keys YA and YB

YA = mod\_exp(alpha, XA, q)

YB = mod\_exp(alpha, XB, q)

# Calculate shared secret keys (private keys) for Alice and Bob

KA = mod\_exp(YB, XA, q) # Alice's private key

KB = mod\_exp(YA, XB, q) # Bob's private key

return YA, YB, KA, KB

def main():

print("Diffie-Hellman Key Exchange")

# User input for prime number q and base alpha

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

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

# User input for public keys XA and XB

XA = int(input("Enter the public key of Alice (XA): "))

XB = int(input("Enter the public key of Bob (XB): "))

# Ensure values are valid for the Diffie-Hellman algorithm

if XA <= 0 or XB <= 0 or XA >= q or XB >= q:

print("Invalid public key values. They must be in the range 1 < XA, XB < q.")

return

# Perform Diffie-Hellman Key Exchange

YA, YB, KA, KB = diffie\_hellman\_key\_exchange(q, alpha, XA, XB)

# Display results

print("\nPublic keys generated:")

print(f"Alice's Public Key (YA): {YA}")

print(f"Bob's Public Key (YB): {YB}")

print("\nPrivate keys calculated:")

print(f"Alice's Private Key (KA): {KA}")

print(f"Bob's Private Key (KB): {KB}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output: