**Experiment No. 6**

**Title:** Diffie-Hellman Key Exchange Protocol

**Batch: B2** **Roll No.: 16010421119**
**Experiment No.: 6**

**Title:** Perform VLab and implement Diffie-Hellman key exchange protocol.

**Resources needed:** Windows/Linux OS

**Theory:**

To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers p and q, such that p is a prime number and q is a generator of p. The generator q is a number that, when raised to positive whole-number powers less than p, never produces the same result for any two such whole numbers. The value of p may be large but the value of q is usually small. Once Alice and Bob have agreed on p and q in private, they choose positive whole-number personal keys a and b, both less than the prime-number modulus p. Neither user divulges their personal key to anyone; ideally they memorize these numbers and do not write them down or store them anywhere. Next, Alice and Bob compute public keys a* and b* based on their personal keys according to the formulas

$$a^* = q^a \bmod p$$

and

$$b^* = q^b \bmod p$$

The two users can share their public keys a* and b* over a communications medium assumed to be insecure, such as the Internet or a corporate wide area network (WAN). From these public keys, a number x can be generated by either user on the basis of their own personal keys. Alice computes x using the formula

$$x = (b^*)^a \bmod p$$

Bob computes x using the formula

$$x = (a^*)^b \bmod p$$

The value of x turns out to be the same according to either of the above two formulas. However, the personal keys a and b, which are critical in the calculation of x, have not been transmitted over a public medium. Because it is a large and apparently random number, a potential hacker has almost no chance of correctly guessing x, even with the help of a powerful computer to conduct millions of trials. The two users can therefore, in theory, communicate privately over a public medium with an encryption method of their choice using the decryption key x.

**Algorithm :**

1) Perform Vlab - https://cse29-iiith.vlabs.ac.in/exp/diffie-hellman/index.html
2) Make use of client-server chatting application and implement following

**A. Client/Sender**

1. Choose a large prime number *p*
2. Calculate generator *g* of *p*
3. Share *p* and *g* with the Server/Receiver

4. Select any natural number (client secrete) *a*
5. *Calculate* $R_A = g^a \bmod p$ and send it to the Server/Receiver
6. Upon receiving $R_B$ from the Server/Receiver, calculate shared key $K_{AB} = (R_B)^a \bmod p$

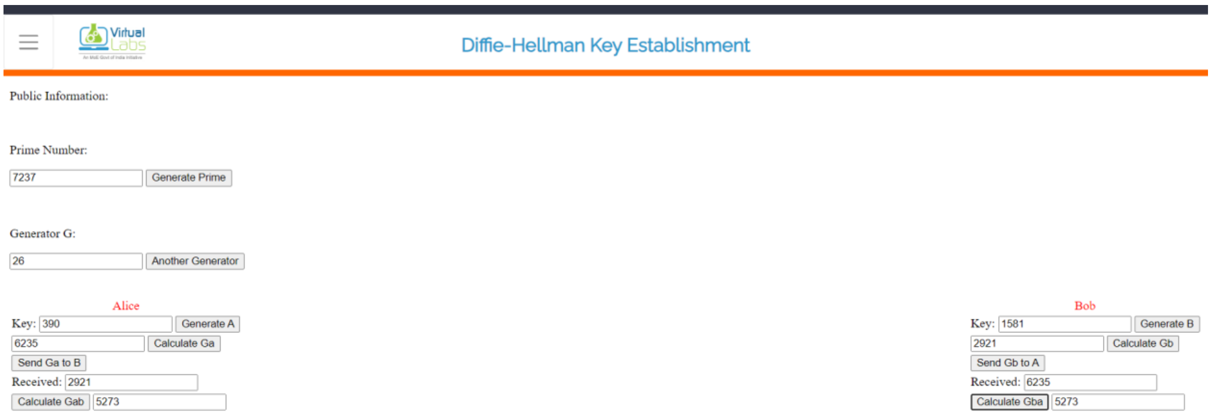
**B. Server/Receiver:**

1. Select any natural number (server secrete) *b*
2. Upon receiving *p* and *g*, calculate $R_B = b^a \bmod p$ and send it to the Client/Sender
3. Upon receiving $R_A$ from the Client/Sender, calculate shared key $K_{AB} = (R_A)^b \bmod p$

**C. NOTE/OBSERVE :** Manually verify that the $K_{AB}$ at both the ends is same.

## Results:

## VLAB:



Diffie-Hellman Key Establishment

Public Information:

Prime Number: 7237 Generate Prime

Generator G: 26 Another Generator

Alice
Key: 390 Generate A
6235 Calculate Ga
Send Ga to B
Received: 2921
Calculate Gab 5273

Bob
Key: 1581 Generate B
2921 Calculate Gb
Send Gb to A
Received: 6235
Calculate Gba 5273

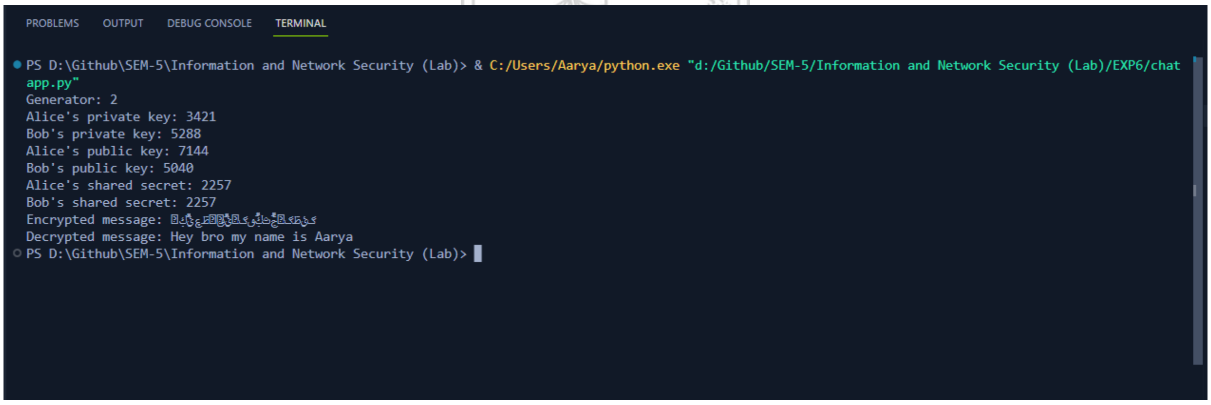
## Code:

```python
import random

def calculate_generator(p):
    for g in range(2, p):
        if pow(g, p - 1, p) == 1:
            return g

def diffie_hellman_key_exchange(p, g, a, RB):
    RA = pow(g, a, p)
    KAB = pow(RB, a, p)

    return RA, KAB
```

```
def xor_encrypt(message, key):
    encrypted_message = ""
    for i in range(len(message)):
        encrypted_char = chr(ord(message[i]) ^ key)
        encrypted_message += encrypted_char
    return encrypted_message

def calculate_modular_exponentiation(q, a, p):
    a_star = pow(q, a, p)
    return a_star


def xor_decrypt(encrypted_message, key):
    decrypted_message = ""
    for i in range(len(encrypted_message)):
        decrypted_char = chr(ord(encrypted_message[i]) ^ key)
        decrypted_message += decrypted_char
    return decrypted_message

def diffie_hellman():
    prime = 7297
    generator = calculate_generator(prime)
    print("Generator:", generator)

    alice_private_key = random.randint(1, prime-1)
    bob_private_key = random.randint(1, prime-1)

    print("Alice's private key:", alice_private_key)
    print("Bob's private key:", bob_private_key)

    alice_public_key = calculate_modular_exponentiation(generator,
alice_private_key, prime)
    print("Alice's public key:", alice_public_key)
    bob_public_key = calculate_modular_exponentiation(generator,
bob_private_key, prime)
    print("Bob's public key:", bob_public_key)

    alice_shared_secret = calculate_modular_exponentiation(bob_public_key,
alice_private_key, prime)
    bob_shared_secret = calculate_modular_exponentiation(alice_public_key,
bob_private_key, prime)

    return alice_shared_secret, bob_shared_secret

alice_shared_secret, bob_shared_secret = diffie_hellman()

print("Alice's shared secret:", alice_shared_secret)
```

```python
print("Bob's shared secret:", bob_shared_secret)

# Message to be sent from Alice to Bob
message = "Hey bro my name is Aarya"

# Encrypt the message using XOR encryption and Alice's shared secret
encrypted_message = xor_encrypt(message, alice_shared_secret)

print("Encrypted message:", encrypted_message)

# Decrypt the message using XOR decryption and Bob's shared secret
decrypted_message = xor_decrypt(encrypted_message, bob_shared_secret)

print("Decrypted message:", decrypted_message)
```

**Output:**



```
PS D:\Github\SEM-5\Information and Network Security (Lab)> & C:/Users/Aarya/python.exe "d:/Github/SEM-5/Information and Network Security (Lab)/EXP6/chat
app.py"
Generator: 2
Alice's private key: 3421
Bob's private key: 5288
Alice's public key: 7144
Bob's public key: 5040
Alice's shared secret: 2257
Bob's shared secret: 2257
Encrypted message: [illegible]
Decrypted message: Hey bro my name is Aarya
PS D:\Github\SEM-5\Information and Network Security (Lab)>
```

---

**Questions:**

1. **Explain any one attack on Diffie-Hellman key exchange protocol.**

   **Ans:**

   One attack that targets the Diffie-Hellman key exchange protocol is the "Man-in-the-Middle" (MitM) attack. The Diffie-Hellman protocol is designed to allow two parties to establish a shared secret key over an insecure communication channel, which can then be used for encrypted communication. However, the MitM attack exploits the vulnerability in the communication channel by inserting an attacker between the two legitimate parties.

**Outcomes:**

**CO3: Describe various access control policies and models.**

---

**Conclusion:**

We can conclude that we have learnt about Diffie-Hellman Key Exchange Protocol.

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

---

**References:**

**Books/ Journals/ Websites:**

- Mark Stamp, "Information security Principles and Practice" Wiley.