**Experiment No. 6**

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

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

**Title:** Perform VLab andimplement 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\* = qa mod p

and

b\* = qb mod 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 mod p

Bob computes x using the formula

x = (a\*)b mod 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.

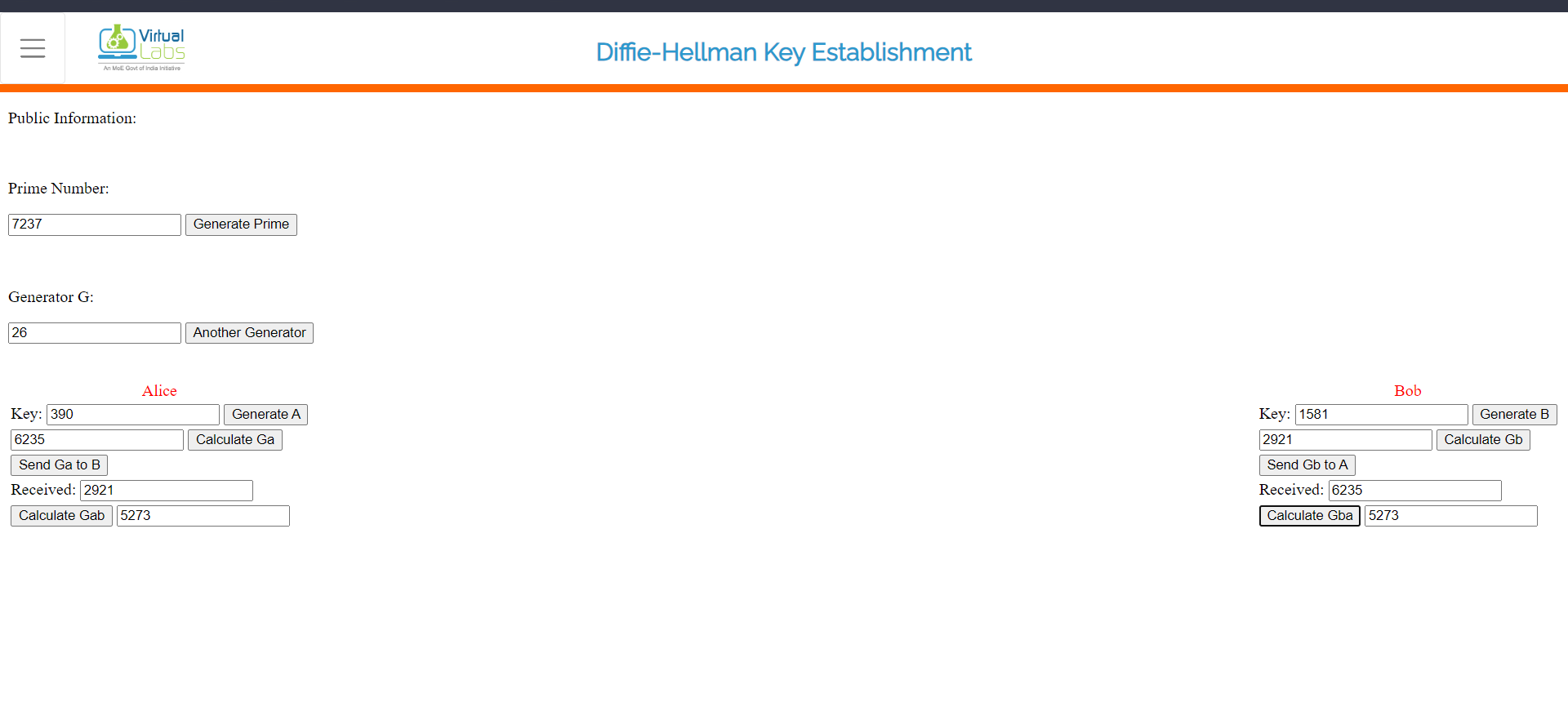
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm :**

1. Perform Vlab - https://cse29-iiith.vlabs.ac.in/exp/diffie-hellman/index.html
2. Make use of client-server chatting applicationand implement following
3. **Client/Sender**
4. Choose a large prime number *p*
5. Calculate generator *g* of *p*
6. Share *p* and *g* with the Server/Receiver
7. Select any natural number (client secrete) *a*
8. *Calculate RA = ga mod p* and send it to the Server/Receiver
9. Upon receiving RB from the Server/Receiver, calculate shared key KAB = (RB )a *mod* *p*
10. **Server/Receiver:**
11. Select any natural number (server secrete) *b*
12. Upon receiving *p* and *g*, calculate *RB = ba mod p* and send it to the Client/Sender
13. Upon receiving RA from the Client/Sender, calculate shared key KAB = (RA )b *mod* *p*
14. **NOTE/OBSERVE :** Manually verify that theKAB at both the ends is same.

**Results:**

**VLAB:**

****

**Code:**

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)

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)

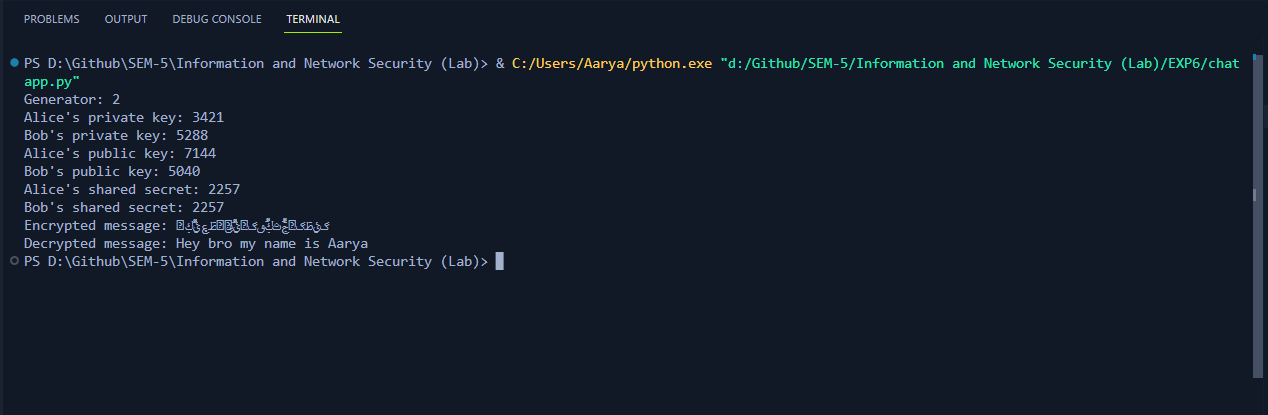
print("Encrypted message:", encrypted\_message)

*# 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:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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.