**Experiment No. 8**

**Aim:** To implement Diffie Hellman Algorithm in python and virtual lab.

**Theory:**

Whitefield Diffie and Martin Hellman develop Diffie Hellman key exchange Algorithms in 1976 to overcome the problem of key agreement and exchange. It enables the two parties who want to communicate with each other to agree on a symmetric key, a key that can be used for encrypting and decryption; Diffie Hellman key exchange algorithm can be used for only key exchange, not for encryption and decryption process. The algorithm is based on mathematical principles.

**Diffie Hellman Algorithm**

1. Shared values

* is a prime number
* is a generator.

2. Key generation for user A

* Select a Private key    Here,

Now, Calculation of Public key

3. Key generation for user B

* Select a Private key    Here,
* Now, Calculation of Public key

4. Calculation of Secret Key by A



5. Calculation of Secret Key by B



**Example**

   = 36,

**Implementation:**

import random

import math

def check\_multiplicative\_inverse(x,y):

if not y:

return x

return check\_multiplicative\_inverse(y,x%y)

class DH:

def \_\_init\_\_(self,prime):

assert self.check\_prime(prime), "Number is not prime"

self.prime = prime

self.relative\_primes = []

for x in range(2,self.prime):

if check\_multiplicative\_inverse(self.prime,x) == 1:

self.relative\_primes.append(x)

self.generator = random.choice(self.relative\_primes)

self.alice\_R\_one()

self.bob\_R\_two()

self.alice\_secret\_key()

self.bob\_secret\_key()

def check\_prime(self,prime):

if prime == 1:

return False

if prime == 2:

return True

if not prime%2:

return False

for x in range(3,math.ceil(math.sqrt(prime))):

if not prime%x:

return False

return True

def alice\_R\_one(self):

self.x = random.randint(0,self.prime)

self.R\_one = pow(self.generator,self.x,self.prime)

def bob\_R\_two(self):

self.y = random.randint(0,self.prime)

self.R\_two = pow(self.generator,self.y,self.prime)

def alice\_secret\_key(self):

self.key\_A = pow(self.R\_two,self.x,self.prime)

def bob\_secret\_key(self):

self.key\_B = pow(self.R\_one,self.y,self.prime)

obj = DH(int(input("Please Enter a prime Number: ")))

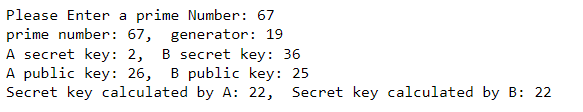
print(f"prime number: {obj.prime}, generator: {obj.generator}\n\

A secret key: {obj.x}, B secret key: {obj.y}\n\

A public key: {obj.R\_one}, B public key: {obj.R\_two}\n\

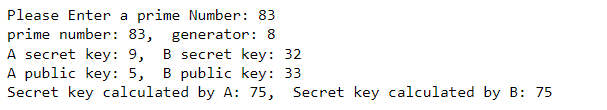
Secret key calculated by A: {obj.key\_A}, Secret key calculated by B: {obj.key\_B}")

**Output:**









**Vlab Output:**

