Practical No. 10

DH Key Exchange

**Q.1]** Write a Python program to demonstrate brute attack, man-in-middle attack, and Diffie-Hellman key exchange.

**Note:** Use the same values as in the output for the program to provide a viable output.

**Code:**

import random

whichAttack = input("Enter which attack (brute/mim/dh): ").lower()

if whichAttack == "brute":

*# BRUTEFORCE ATTACK*

prime = int(input("Enter primitive root you got from unsecured channel: "))

mod = int(input("Enter prime number which you got from unsecured channel: "))

public\_alice = int(input("Enter public key of alice you eavesdropped from insecure channel: "))

public\_bob = int(input("Enter public key of bob you eavesdropped from insecure channel: "))

possible\_private\_keys = []

print("This below keys could be the private key of alice: ")

for i in range(100):

guess = prime \*\* i % mod

if guess == public\_alice:

possible\_private\_keys.append(i)

print(possible\_private\_keys)

possible\_private\_keys\_bob = []

print("This below keys could be the private key of Bob : ")

for i in range(100):

guess = prime \*\* i % mod

if guess == public\_bob:

possible\_private\_keys\_bob.append(i)

print(possible\_private\_keys\_bob)

alice\_guess = 0

bob\_guess = 0

for i in possible\_private\_keys:

alice\_guess = (public\_bob \*\* i) % mod

for i in possible\_private\_keys\_bob:

bob\_guess = (public\_alice \*\* i) % mod

if alice\_guess == bob\_guess:

print("Attack successful")

print("THIS COULD BE YOUR SHARED SECRET KEY WITH ALICE: ", alice\_guess)

print("THIS COULD BE YOUR SHARED SECRET KEY WITH BOB: ", bob\_guess)

print("Your shared secret key is: ", alice\_guess)

elif whichAttack == "mim":

*# Diffie-Hellman Key Exchange*

*# Shared prime number and primitive*

prime = 23

primitive = 5

*# Alice's private key*

alice\_private\_key = 20

*# Bob's private key*

bob\_private\_key = 10

*# Calculate public keys*

alice\_public\_key = (primitive \*\* alice\_private\_key) % prime

bob\_public\_key = (primitive \*\* bob\_private\_key) % prime

*# Calculate shared secret*

alice\_shared\_secret = (bob\_public\_key \*\* alice\_private\_key) % prime

bob\_shared\_secret = (alice\_public\_key \*\* bob\_private\_key) % prime

print("Alice's Public Key:", alice\_public\_key)

print("Bob's Public Key:", bob\_public\_key)

print("Shared Secret (Alice):", alice\_shared\_secret)

print("Shared Secret (Bob):", bob\_shared\_secret)

else:

p = int(input('Enter a prime number (p): '))

g = int(input('Enter a base number (g): '))

*# Private numbers*

*# a = random.randint(1, p)*

*# b = random.randint(1, p)*

*# c = random.randint(1, p)*

*# d = random.randint(1, p)*

a = 3

b = 2

c = 1

d = 4

*# Public values*

ga = (g \*\* a) % p

gb = (g \*\* b) % p

gc = (g \*\* c) % p

gd = (g \*\* d) % p

*# Computing shared secret keys*

s1 = (gd \*\* a) % p

s2 = (gc \*\* b) % p

print(**f**"Alice's private number (a): {a}")

print(**f**"Bob's private number (b): {b}")

print(**f**"Malory's private number for Alice (c): {c}")

print(**f**"Malory's private number for Bob (d): {d}")

print(**f**"Alice's public value (ga): {ga}")

print(**f**"Bob's public value (gb): {gb}")

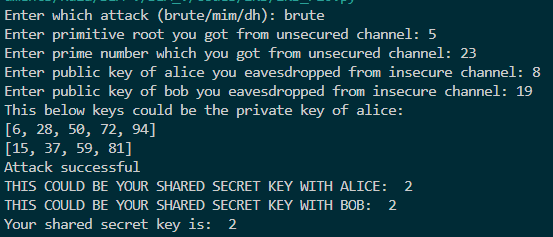
print(**f**"Malory's manipulated value for Alice (gc): {gc}")

print(**f**"Malory's manipulated value for Bob (gd): {gd}")

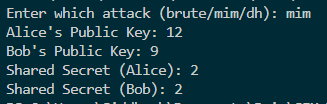
print(**f**"Alice's computed shared secret key (S1): {s1}")

print(**f**"Bob's computed shared secret key (S2): {s2}")

**Output:**



**DH key exchange**

****

**MIM**

