Final Year B. Tech. (CSE) – I: 2022-23

4CS451: Cryptography and Network Security Lab

Assignment No. 11

PRN: 2019BTECS00077 **Batch: B7**

Full name: Biradar Avinash Vishnu

Title: Diffie Helman key exchange Algorithm Implementation

Objective: securely exchange the key between sender and receiver.

Code:

Server.py:

```
import socket
import os
import sys
from sympy import isprime
import math


# Python3 program to find primitive root
# of a given number n
# from math import sqrt

""" Iterative Function to calculate (x^n)%p
    in O(logy) */"""
def power( x, y, p):

    res = 1 # Initialize result

    x = x % p # Update x if it is more
            # than or equal to p

    while (y > 0):
```

```
        # If y is odd, multiply x with result
        if (y & 1):
            res = (res * x) % p

        # y must be even now
        y = y >> 1 # y = y/2
        x = (x * x) % p

    return res

# Utility function to store prime
# factors of a number
def findPrimefactors(s, n) :

    # Print the number of 2s that divide n
    while (n % 2 == 0) :
        s.add(2)
        n = n // 2

    # n must be odd at this point. So we can
    # skip one element (Note i = i +2)
    for i in range(3, int(math.sqrt(n)), 2):

        # While i divides n, print i and divide n
        while (n % i == 0) :

            s.add(i)
            n = n // i

    # This condition is to handle the case
    # when n is a prime number greater than 2
    if (n > 2) :
        s.add(n)

# Function to find smallest primitive
# root of n
def findPrimitive( n) :
    s = set()

    # Check if n is prime or not
    if (isprime(n) == False):
        return -1

    # Find value of Euler Totient function
    # of n. Since n is a prime number, the
```

```
    # value of Euler Totient function is n-1
    # as there are n-1 relatively prime numbers.
    phi = n - 1

    # Find prime factors of phi and store in a set
    findPrimefactors(s, phi)

    # Check for every number from 2 to phi
    for r in range(2, phi + 1):

        # Iterate through all prime factors of phi.
        # and check if we found a power with value 1
        flag = False
        for it in s:

            # Check if r^((phi)/primefactors)
            # mod n is 1 or not
            if (power(r, phi // it, n) == 1):

                flag = True
                break

        # If there was no power with value 1.
        if (flag == False):
            return r

    # If no primitive root found
    return -1

# Driver Code



# This code is contributed by
# Shubham Singh(SHUBHAMSINGH10)




print("**SERVER PROGRAM STARTED ***")
s=socket.socket()
#host=socket.gethostname()
host='127.0.0.1'
port=12000 #ports after 6000 are free
```

```
s.bind((host,port))
s.listen(10)

c,addr=s.accept()
print ("Client connected",addr)
print ('Got Connection from' ,addr)
PublicB=c.recv(100).decode()
print("Received integer",PublicB)

print("Received integer succssfully ")



while True :
    content = (input("Enter a large prime number : "))
    p = int(content)
    if isprime(p):
        PrivateA = int(input("Enter the private Key : "))
        G = findPrimitive(p)
        PublicA = str(int(pow(G,PrivateA,p)))
        c.send(PublicA.encode())   # this is integer
        # print(PublicA)
        ka = int(pow(int(PublicB),PrivateA,p))
        print("Secret Key Of A : ",ka)
        break
    else :
        print("Its not a prime number")
print("Bye")




print("**SERVER PROGRAM ENDED ***")
s.close()
```

Client.py:

```
import socket
import os
import math
import sympy
```

```
""" Iterative Function to calculate (x^n)%p
    in O(logy) */"""


def power(x, y, p):

    res = 1  # Initialize result

    x = x % p  # Update x if it is more
    # than or equal to p

    while (y > 0):

        # If y is odd, multiply x with result
        if (y & 1):
            res = (res * x) % p

        # y must be even now
        y = y >> 1  # y = y/2
        x = (x * x) % p

    return res

# Utility function to store prime
# factors of a number


def findPrimefactors(s, n):

    # Print the number of 2s that divide n
    while (n % 2 == 0):
        s.add(2)
        n = n // 2

    # n must be odd at this point. So we can
    # skip one element (Note i = i +2)
    for i in range(3, int(math.sqrt(n)), 2):

        # While i divides n, print i and divide n
        while (n % i == 0):

            s.add(i)
            n = n // i

    # This condition is to handle the case
    # when n is a prime number greater than 2
    if (n > 2):
        s.add(n)

# Function to find smallest primitive
# root of n


def findPrimitive(n):
    s = set()

    # Check if n is prime or not
    if (sympy.isprime(n) == False):
```

```
        return -1

    # Find value of Euler Totient function
    # of n. Since n is a prime number, the
    # value of Euler Totient function is n-1
    # as there are n-1 relatively prime numbers.
    phi = n - 1

    # Find prime factors of phi and store in a set
    findPrimefactors(s, phi)

    # Check for every number from 2 to phi
    for r in range(2, phi + 1):

        # Iterate through all prime factors of phi.
        # and check if we found a power with value 1
        flag = False
        for it in s:

            # Check if r^((phi)/primefactors)
            # mod n is 1 or not
            if (power(r, phi // it, n) == 1):

                flag = True
                break

        # If there was no power with value 1.
        if (flag == False):
            return r

    # If no primitive root found
    return -1


print("*****CLIENT PROGRAM STARTED *******")
s = socket.socket()
# host=socket.gethostname() #server hostname
host = '127.0.0.1'
port = 12000  # same as server
s.connect((host, port))

print("Connected to : ", host, port)

# getting input from client
# content=1
while True:
    content = (input("Enter a large prime number : "))
    p = int(content)
    if sympy.isprime(p):
        privateB = int(input("Enter the private Key val: "))
        G = findPrimitive(p)
        # print("P : ", p, " privateB: ", privateB, " G", G)
        PublicB = str(int(pow(G, privateB, p)))
        # print(PublicB)
        s.send(PublicB.encode())   # this is integer
        publicA = s.recv(100).decode()
        # print("PublicA : ", publicA)
        kb = int(pow(int(publicA), privateB, p))
```

```
        print("Secret Key Of B: ", kb)
        break
    else:
        print("OOPs that was not a prime try again")

# s.send(content.encode())

# print("Sent successfully")

# print("Received integer", content)

print("*****CLIENT PROGRAM ENDED *******")
s.close()
```

Output:

```
PS E:\College\Final year\C&NS\practical> python client.py
*****CLIENT PROGRAM STARTED *******
Connected to :  127.0.0.1 12000
Enter a large prime number : 97
Enter the private Key val: 23
Secret Key Of B:  31
*****CLIENT PROGRAM ENDED *******
PS E:\College\Final year\C&NS\practical>
```

```
PS E:\College\Final year\C&NS\practical> python server.py
**SERVER PROGRAM STARTED ***
Client connected ('127.0.0.1', 60165)
Got Connection from ('127.0.0.1', 60165)
Received integer 82
Received integer succssfully
Enter a large prime number : 97
Enter the private Key : 2
Secret Key Of A :  31
 ye
**SERVER PROGRAM ENDED ***
```