Deep Nayak: 2019130045
Batch - C TE Comps
09/10/2021

EXPERIMENT 2: Key Exchange Algorithm

Aim: To understand the concept of key exchange algorithms and implement the Diffie Hellman Algorithm in python followed by testing the same with big prime numbers as input.

Code:

```
import random, math
from math import sqrt

def isPrime( n):
    if (n <= 1):
        return False
    if (n <= 3):
        return True
    if (n % 2 == 0 or n % 3 == 0):
        return False
    i = 5
    while(i * i <= n):
        if (n % i == 0 or n % (i + 2) == 0) :
            return False
        i = i + 6
    return True

def power( x, y, p):
    res = 1
    x = x % p
    while (y > 0):
        if (y & 1):
            res = (res * x) % p
        y = y >> 1
        x = (x * x) % p
    return res
```

```
def findPrimefactors(s, n) :
    while (n % 2 == 0) :
        s.add(2)
        n = n // 2
    for i in range(3, int(sqrt(n)), 2):
        while (n % i == 0) :
            s.add(i)
            n = n // i
    if (n > 2) :
        s.add(n)

def findPrimitive( n) :
    s = set()
    if (isPrime(n) == False):
        return -1
    phi = n - 1
    findPrimefactors(s, phi)
    for r in range(2, phi + 1):
        flag = False
        for it in s:
            if (power(r, phi // it, n) == 1):
                flag = True
                break
        if (flag == False):
            return r
    return -1

print("Enter a prime number: ")
primeNumber = int(input())

primitiveRoot = findPrimitive(primeNumber)
print("The primitive root of", primeNumber, "is", primitiveRoot)

print("Enter private key for user A: ")
privateKeyA = int(input())
publicKeyA = power(primitiveRoot, privateKeyA, primeNumber)
print("The public key for user A is", publicKeyA)

print("Enter private key for user B: ")
```

```
privateKeyB = int(input())
publicKeyB = power(primitiveRoot, privateKeyB, primeNumber)
print("The public key for user A is", publicKeyB)

sA = power(publicKeyB, privateKeyA, primeNumber)
sB = power(publicKeyA, privateKeyB, primeNumber)

print("The secret key for user A is", sA)
print("The secret key for user B is", sB)
```

Output:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak$
Enter a prime number:
4093082899
Enter the second prime number:
2860486313
Enter private key for user A:
5915587277
The information shared by user A is 1312879516
Enter private key for user B:
3367900313
The information shared by user B is 1963984909
The secret key for user A is 20893543
The secret key for user B is 20893543
```

Conclusion:

Through this experiment I learnt about different types of key exchange algorithms and implemented the Diffie Hellman algorithm. My findings about the same are as follows:

1. The Diffie Hellman algorithm makes use of primitive roots of any prime numbers that are given by the user. However calculation of the primitive root is a computationally heavy process, hence, instead of taking it as input from the user, I calculated the smallest primitive root of the given prime number. In order to do that I had to write a code for finding the prime factors of a given prime number which was in turn dependent on finding if a given number is prime or not.
2. Since the exponentiation and modulo operation are used together in this algorithm, I had to write a custom power calculation function that accepts 3 numbers (namely the base, the power to which the number is supposed to be raised and the modulo number) and calculates the power using the binary exponentiation algorithm in order to minimize the time complexity.
3. I think the Diffie Hellman algorithm is the simplest and very efficient cryptographic method that uses simple prime factorization concepts to securely generate a

symmetric key over a public network. The implementation can pose a challenge if we are dealing with large prime numbers but it can easily be scaled to use it in real world applications.

Github Link: https://github.com/deepnayak/CSS-Lab-Deep-Nayak