Experiment 2

Diffie Hellman Algorithm

Name: Jash Jain
UID: 2019130021
Class: TE Comps

Objective:
To implement the Diffie Hellman Algorithm

Theory:

Diffie–Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman.DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Published in 1976 by Diffie and Hellman, this is the earliest publicly known work that proposed the idea of a private key and a corresponding public key.

Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical means, such as paper key lists transported by a trusted courier. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric-key cipher.

Diffie–Hellman is used to secure a variety of Internet services. However, research published in October 2015 suggests that the parameters in use for many DH Internet applications at that time are not strong enough to prevent compromise by very well-funded attackers, such as the security services of some countries.

Code:

```
from math import sqrt

def prime_checker(n):
    g_or_not = 0
    if n > 1:
        for i in range(2, int(sqrt(n)) + 1):
            if (n % i == 0):
                g_or_not = 1
                break
        if (g_or_not == 0):
            return True
        else:
            return False
    return False


def diffie_hellman():
    g = int(input('Enter 1st Prime number: '))
    if not prime_checker(g):
        g = int(input("ERR!!! Number not Prime -- Try again: "))
```

```
    n = int(input('Enter the 2nd Prime Number : '))
    if not prime_checker(n):
        n = int(input("ERR!!! Number not Prime -- Try again: "))

    x = int(input('\nSecret X(Alice): '))
    y = int(input('Secret Y(Bob): '))

    X_alice = int(pow(n,x,g))
    X_bob = int(pow(n,y,g))

    print("Value generated by A sent to B is ",X_alice)
    print("Value generated by B sent to A is ",X_bob)

    ka = int(pow(X_bob,x,g))
    kb = int(pow(X_alice,y,g))

    print('\nResults are as follows \nSecret key K1 :',ka)
    print('Secret Key K2 :',kb)

if __name__  == '__main__':
    diffie_hellman()
```

Screenshots:

```
Enter 1st Prime number:
5915587277
Enter the 2nd Prime Number :
9576890767

Secret X(Alice):
2379301732
Secret Y(Bob):
9842084289
Value generated by A sent to B is  455830886
Value generated by B sent to A is  2819057864

Results are as follows
Secret key K1 : 4408553447
Secret Key K2 : 4408553447



** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Conclusion:

Through this experiment, I learned about different types of key exchange algorithms and implemented the Diffie Hellman algorithm. My findings of 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. The process of checking whether a number is prime or not requires more computation time as the prime number can be very big also.The beauty of this algorithm is that the secret key gets generated on both sides without exchanging the key.

Github Link:
https://github.com/jashjain21/CSS_LAB