Ishita Hardasmalani

C14-2103058

EXPERIMENT NO. 6

Aim: Write a program to implement Diffie-Hellman Algorithm

Theory:

The Diffie-Hellman key exchange, named after its inventors Whitfield Diffie and Martin Hellman, is a method for two parties to establish a shared secret key over an insecure channel. This shared secret key can then be used for secure communication using symmetric encryption techniques. Diffie-Hellman is a key exchange algorithm rather than an encryption algorithm itself.

The security of the Diffie-Hellman key exchange relies on the difficulty of solving the discrete logarithm problem, which involves finding the exponent (private key) given the base, modulus, and result (public key). Even if an eavesdropper intercepts the public keys exchanged during the key exchange, it is computationally infeasible for them to derive the shared secret key without knowledge of the private keys.

Diffie-Hellman key exchange is widely used in various cryptographic protocols, including secure internet communication (such as TLS/SSL) and secure messaging protocols. It provides a secure method for establishing a shared secret key over an insecure channel, enabling secure communication between parties.

Working:

1. Setup:

- Choose a large prime number, p, and a primitive root modulo p, g. These values are public and can be agreed upon in advance or exchanged openly.
- Each party also generates its private key:
  - Party A chooses a private key a randomly from the range [1, p-1].
  - Party B chooses a private key b randomly from the range [1, p-1].

2. Key Exchange:

- Party A computes its public key, A, using the formula: A = g^a mod p.

- Party B computes its public key, B, using the formula: B = g^b mod p.

- Party A sends its public key, A, to Party B.

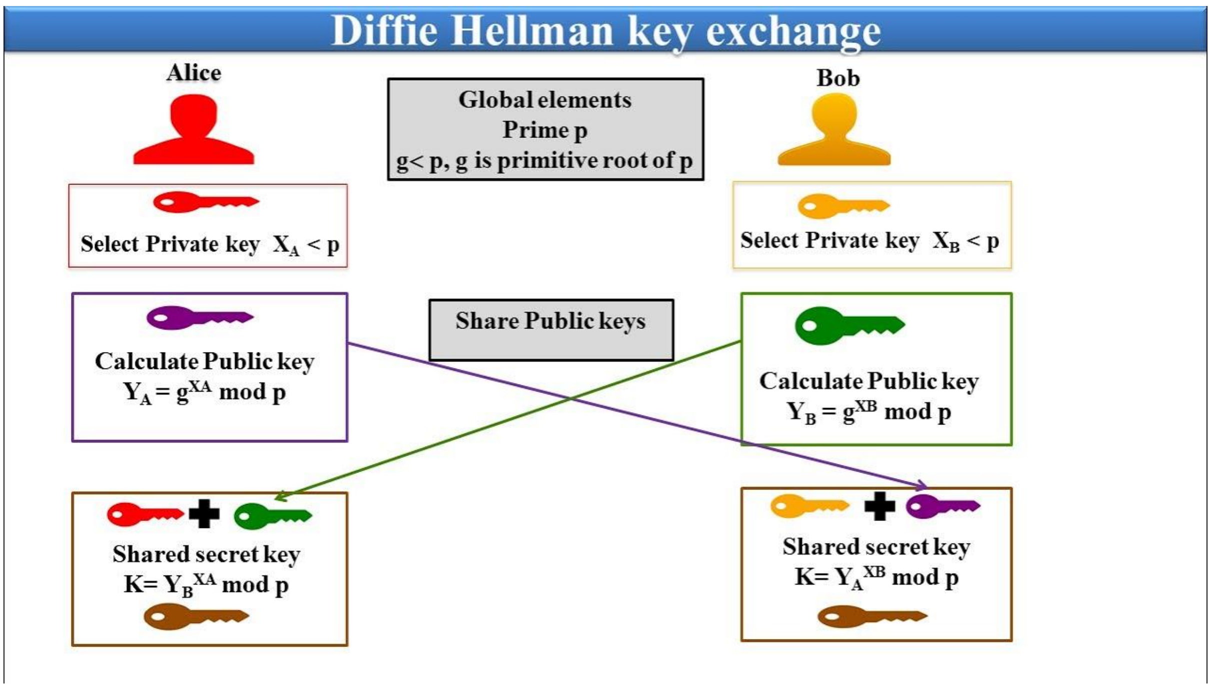
- Party B sends its public key, B, to Party A.

3. Secret Key Derivation:

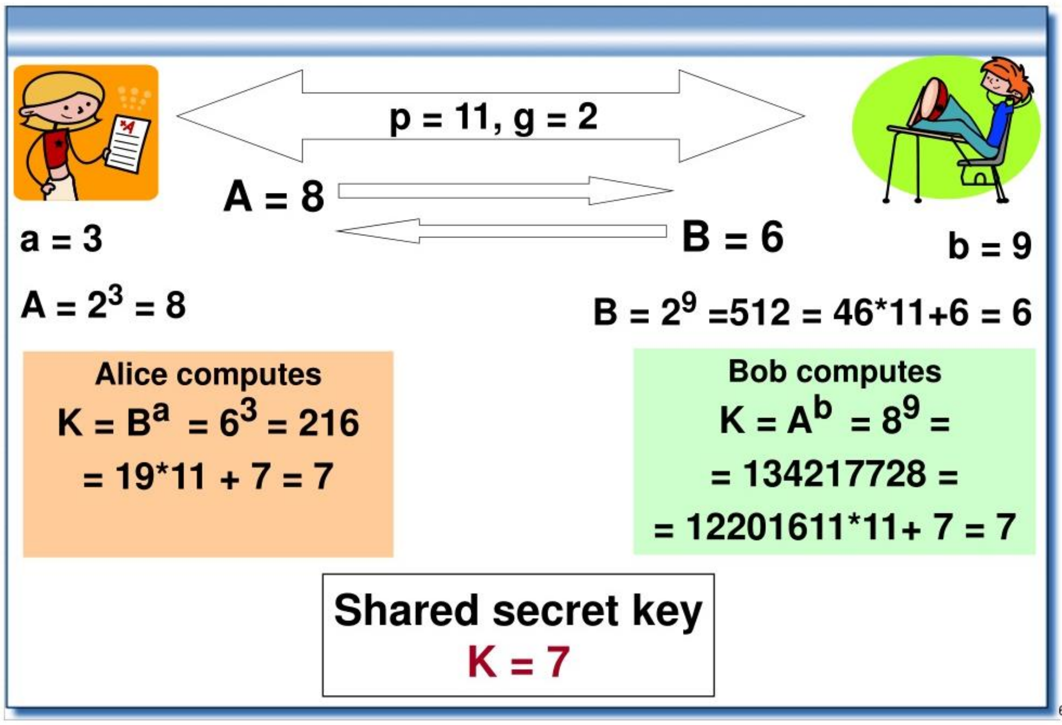
- Party A computes the shared secret key using Party B's public key: K = B^a mod p.

- Party B computes the shared secret key using Party A's public key: K = A^b mod p.

- Both parties now have the same shared secret key, K, which they can use for secure communication using symmetric encryption algorithms like AES.



Example:



$A = 2^3 = 8$

$B = 2^9 = 512 = 46*11+6 = 6$

Alice computes

$K = B^a = 6^3 = 216$

$= 19*11 + 7 = 7$

Bob computes

$K = A^b = 8^9 =$

$= 134217728 =$

$= 12201611*11 + 7 = 7$

[6] The steps are as follows:

1. Alice chooses a large random number x such that $0 \le x \le p - 1$ and calculates

$$R1 = g^x \bmod p.$$

2. Bob chooses another large random number y such that $0 \le y \le p - 1$ and calculates

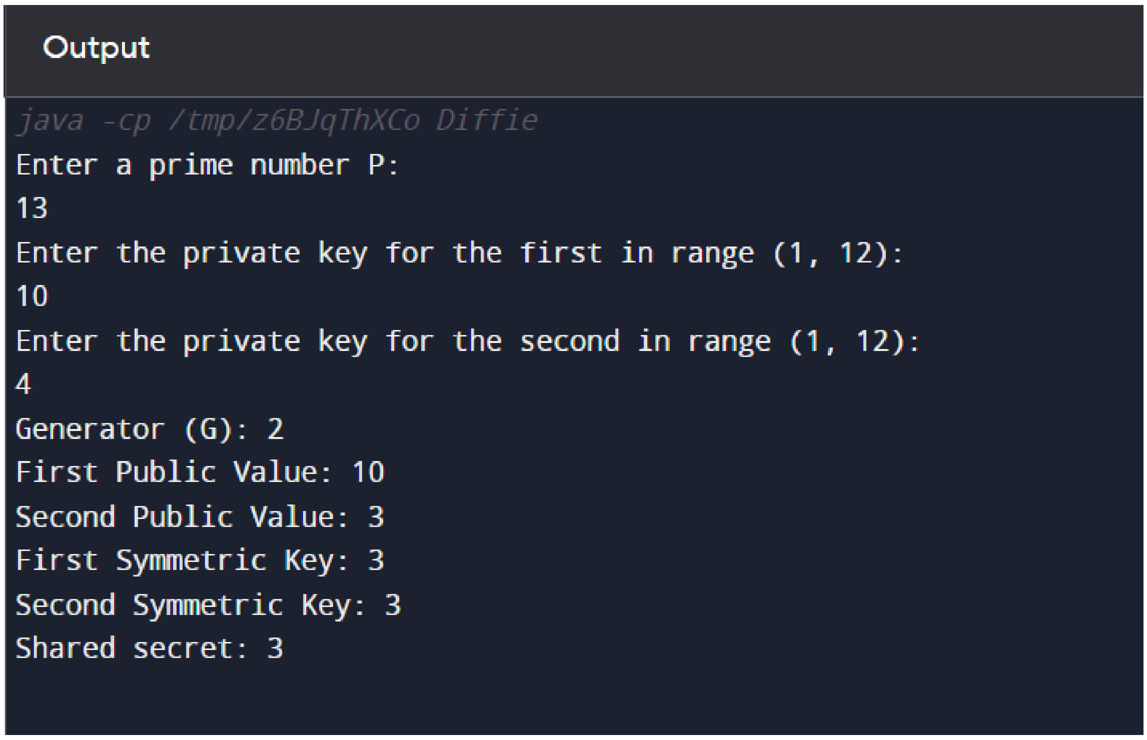
$$R2 = g^y \bmod p.$$

3. Alice sends R1 to Bob. Note that Alice does not send the value of x; she sends only R1.
4. Bob sends R2 to Alice. Again, note that Bob does not send the value of y, he sends only R2.
5. Alice calculates $K = (R2)^x \bmod p$.
6. Bob also calculates $K = (R1)^y \bmod p$.

Code:

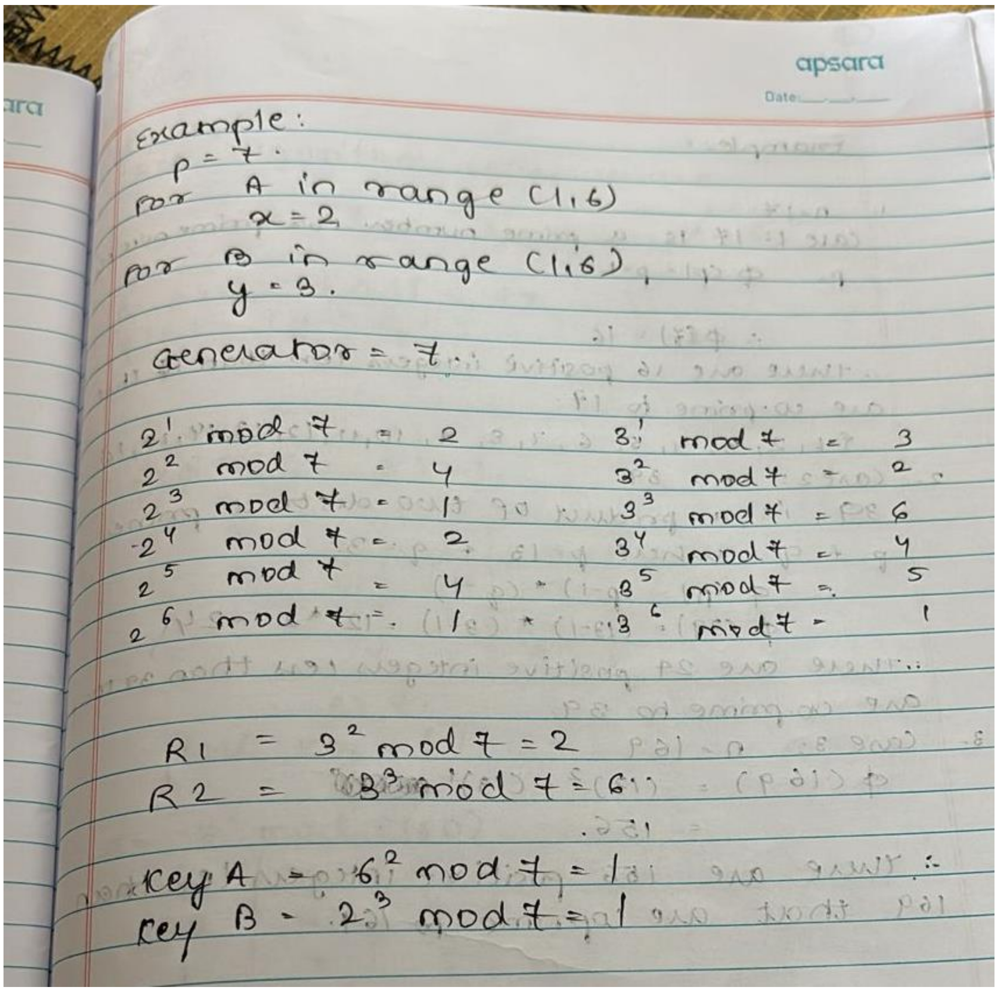
```
import java.util.Scanner;

public class Diffie {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int P;
        do {
            System.out.println("Enter a prime number P:");
            P = scanner.nextInt();
            if (!isPrime(P)) {
                System.out.println("Number is not prime, please enter a prime
number.");
            }
        } while (!isPrime(P));

        int privateKey1 = getPrivateKey(scanner, P, "first");
        int privateKey2 = getPrivateKey(scanner, P, "second");

        int G = findGenerator(P);
        System.out.println("Generator (G): " + G);

        int publicValue1 = computePublicValue(G, privateKey1, P);
        int publicValue2 = computePublicValue(G, privateKey2, P);
        System.out.println("First Public Value: " + publicValue1);
        System.out.println("Second Public Value: " + publicValue2);

        int key1 = computePublicValue(publicValue2, privateKey1, P);
        int key2 = computePublicValue(publicValue1, privateKey2, P);

        System.out.println("First Symmetric Key: " + key1);
        System.out.println("Second Symmetric Key: " + key2);

        if (key1 == key2) {
            System.out.println("Shared secret: " + key1);
        } else {
            System.out.println("Something went wrong. The keys do not
match.");
        }
    }

    private static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }

    private static int getPrivateKey(Scanner scanner, int P, String user) {
        int key;
        do {
            System.out.println("Enter the private key for the " + user + " in
range (1, " + (P - 1) + "):");
            key = scanner.nextInt();
            if (key <= 1 || key >= P) {
                System.out.println("Key is out of range, please enter a valid
key.");
            }
        } while (key <= 1 || key >= P);
        return key;
    }

    private static int findGenerator(int P) {
        for (int potentialG = 2; potentialG < P; potentialG++) {
            boolean isGenerator = true;
            for (int power = 1; power < P - 1; power++) {
                if (Math.pow(potentialG, power) % P == 1 && power < P - 1) {
                    isGenerator = false;
                    break;
                }
            }
            if (isGenerator) return potentialG;
        }
        return -1;
    }

    private static int computePublicValue(int base, int exponent, int
modulus) {
        return (int) Math.pow(base, exponent) % modulus;
    }
}
```

Output:



```
Output
java -cp /tmp/z6BJqThXCo Diffie
Enter a prime number P:
13
Enter the private key for the first in range (1, 12):
10
Enter the private key for the second in range (1, 12):
4
Generator (G): 2
First Public Value: 10
Second Public Value: 3
First Symmetric Key: 3
Second Symmetric Key: 3
Shared secret: 3
```

Example:



Example:

$p = 7$.

For A in range (1,6)

$x = 2$

For B in range (1,6)

$y = 3$.

Generator = 7.

$2^1 \bmod 7 = 2$	$3^1 \bmod 7 = 3$
$2^2 \bmod 7 = 4$	$3^2 \bmod 7 = 2$
$2^3 \bmod 7 = 1$	$3^3 \bmod 7 = 6$
$2^4 \bmod 7 = 2$	$3^4 \bmod 7 = 4$
$2^5 \bmod 7 = 4$	$3^5 \bmod 7 = 5$
$2^6 \bmod 7 = 1$	$3^6 \bmod 7 = 1$

$R1 = 3^2 \bmod 7 = 2$

$R2 = 3^3 \bmod 7 = 6$.

Key A $= 6^2 \bmod 7 = 1$

Key B $= 2^3 \bmod 7 = 1$