

# Diffie-Hellmann key exchange

## Aim:

To program a java code for implementation of Diffie Hellmann key exchange algorithm.

## Algorithm:

Step 1: Choose two prime numbers  **$g$ (primitive root of  $p$ )** and  **$p$** .

Step 2: Alice selects a secret no( **$a$** ) and computes  **$g^a \bmod p$** , let's call it  **$A$** . Alice sends  **$A$**  to Bob.

Step 3: Bob selects a secret no( **$b$** ) and computes  **$g^b \bmod p$** , let's call it  **$B$** . Bob sends  **$B$**  to Alice.

Step 4: Alice computes  **$S_A = B^a \bmod p$**

Step 5: Bob computes  **$S_B = A^b \bmod p$**

Step 6: If  **$S_A = S_B$**  then Alice and Bob can agree for future communication.

## Resources Used:

Java 8 and Windows/Linux

## About Diffie-Hellman key exchange:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b. P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

### Step by Step Explanation:

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a \text{ mod } P$	Key generated = $y = G^b \text{ mod } P$
Exchange of generated keys takes place	

Key received = y	Key received = x
Generated Secret Key = $k_a = y^a \bmod P$	Generate Secret Key = $k_b = x^b \bmod P$
Algebraically, it can be shown that $k_a = k_b$	

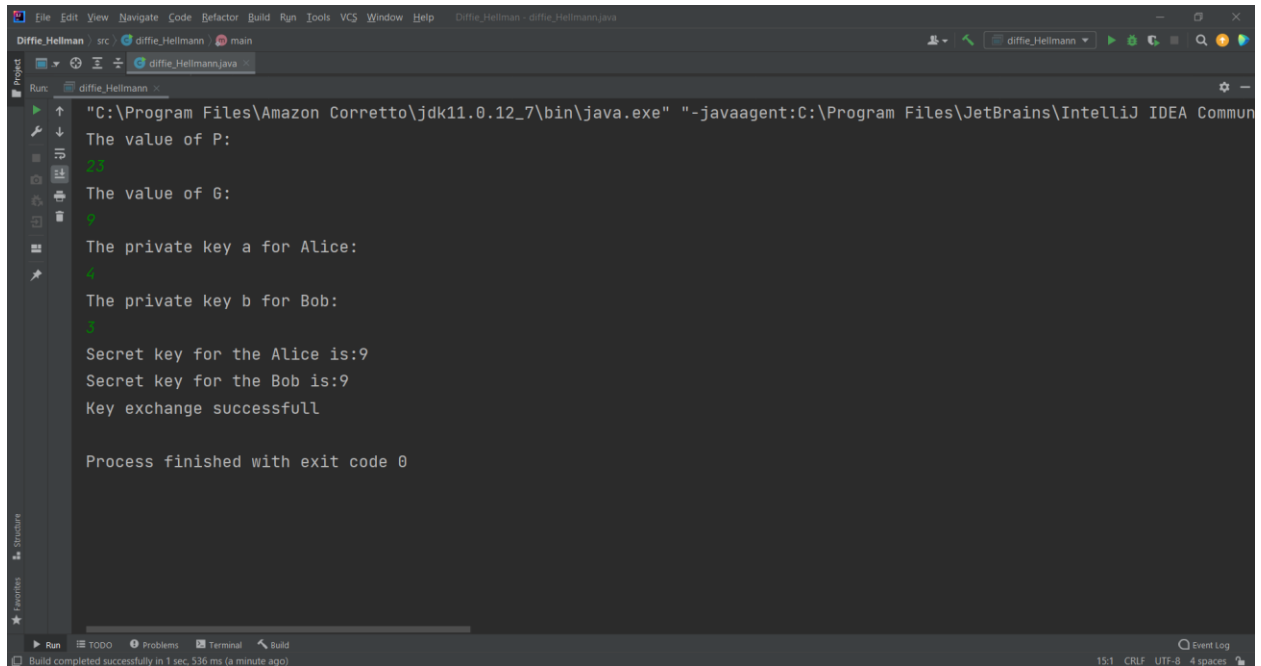
## Code Screenshot:

```

1  public class diffie_Hellmann {
2      private static long power(long a, long b, long p){
3          if (b == 1)
4              return a;
5          else
6              return (((long)Math.pow(a, b)) % p);
7      }
8
9      public static void main(String[] args){
10         Scanner sc = new Scanner(System.in);
11         long P, G, x, a, y, b, ka, kb;
12         System.out.println("The value of P:");
13         P = sc.nextInt();
14         System.out.println("The value of G:");
15         G = sc.nextInt();
16         System.out.println("The private key a for Alice:");
17         a = sc.nextInt();
18         x = power(G, a, P);
19         System.out.println("The private key b for Bob:");
20         b = sc.nextInt();
21         y = power(G, b, P);
22         ka = power(y, a, P);
23         kb = power(x, b, P);
24         System.out.println("Secret key for the Alice is: " + ka);
25         System.out.println("Secret key for the Bob is: " + kb);
26         if (ka == kb){
27             System.out.println("Key exchange successful");
28         }
29         else {
30             System.out.println("Key Exchange unsuccessful");
31         }
32     }
33 }

```

## Code Output:



```
"C:\Program Files\Amazon Corretto\jdk11.0.12_7\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commun
The value of P:
23
The value of G:
6
The private key a for Alice:
12
The private key b for Bob:
12
Secret key for the Alice is:9
Secret key for the Bob is:9
Key exchange successfull

Process finished with exit code 0
```

## GitHub Link:

[https://github.com/AyushGupta-Code/Diffie\\_Hellmann.git](https://github.com/AyushGupta-Code/Diffie_Hellmann.git)