Laboratory_08: Implementation of Diffie-Hellman Algorithm

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. Step-by-Step explanation is as follows:

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a mod P$	Key generated = $y = G^b mod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key = $k_b = x^b mod P$
Algebraically, it can be shown that $k_a = k_b$	
Users now have a symmetric secret key to encrypt	

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9
Step 2: Alice selected a private key a = 4 and
Bob selected a private key b = 3
Step 3: Alice and Bob compute public values
Alice: x =(9^4 mod 23) = (6561 mod 23) = 6
Bob: y = (9^3 mod 23) = (729 mod 23) = 16
Step 4: Alice and Bob exchange public numbers
Step 5: Alice receives public key y =16 and
Bob receives public key x = 6
Step 6: Alice and Bob compute symmetric keys
Alice: ka = y^a mod p = 65536 mod 23 = 9
Bob: kb = x^b mod p = 216 mod 23 = 9
Step 7: 9 is the shared secret.

Python code:

```
# Diffie-Hellman Code

# Power function to return value of a^b mod P

def power(a, b, p):
    if b == 1:
        return a
    else:
        return pow(a, b) % p

# Main function
def main():
    # Both persons agree upon the public keys G and P
    # A prime number P is taken
    P = 23
    print("The value of P:", P)

    # A primitive root for P, G is taken
    G = 9
    print("The value of G:", G)
```

```
    # Alice chooses the private key a
    # a is the chosen private key
    a = 4
    print("The private key a for Alice:", a)

    # Gets the generated key
    x = power(G, a, P)

    # Bob chooses the private key b
    # b is the chosen private key
    b = 3
    print("The private key b for Bob:", b)

    # Gets the generated key
    y = power(G, b, P)

    # Generating the secret key after the exchange of keys
    ka = power(y, a, P)  # Secret key for Alice
    kb = power(x, b, P)  # Secret key for Bob

    print("Secret key for Alice is:", ka)
    print("Secret key for Bob is:", kb)

if __name__ == "__main__":
    main()
```