# Diffie-Hellman key exchange algorithm

Hugo Lamballais and Vianney de Maximy

Introduction :

The Diffie-Hellman Key Exchange Algorithm is a cryptographic method that enables two different parties (the parties representing 2 sides that want to share a key) to securely establish a shared secret over an insecure communication channel. We can name the first person Alice and the second one Bob. The process works with these steps :

1. Both parties agree on a large prime number p and a primitive root g modulo p, which are public parameters.
2. Each party generates a private key: a for Alice and b for Bob, kept secret.
3. Alice computes her public key as $A = g^a \bmod p$ and Bob computes $B = g^b \bmod p$.
4. Alice and Bob exchange their public keys A and B.
5. Each party uses the other's public key to compute the shared secret:
    - Alice computes $S = B^a \bmod p$,
    - Bob computes $S = A^b \bmod p$.
6. Both compute the same shared secret $S = g^{ab} \bmod p$ due to the properties of modular arithmetic.

This shared secret can then be used as a key for symmetric encryption, enabling secure communication.

Short description of the algorithm :
In our code, the user has to choose in the menu to change the input parameters if he wishes to change them from default which was, for us, p=23 and g=11. He then chooses two secret keys and the code will print out all the calculations (it can be removed, we just thought it was better with the calculations printed out) and then return $B^a \bmod p$ as result[0] and $A^b \bmod p$ as result[1].
Then, in the main file, we verify if they are equal (if the exchange is successful) and if it's the case, we return the shared key.

Tests :
For p=23 and m=11 :

```
The parameters are p = 23 and g =  11
Do you want to change the parameters ? 0 : no, 1 : yes 0
Enter your Secret key 5
11 ** 5  modulo  23  =  5
Enter your Secret key 9
11 ** 9  modulo  23  =  19
So the shared key will be :
5 ** 9  modulo  23  =  11
19 ** 5  modulo  23  =  11
Key exchange successful. Shared key is: 11
```

We try again with other secrets :

```
The parameters are p = 23 and g =  11
Do you want to change the parameters ? 0 : no, 1 : yes 0
Enter your Secret key 6
11 ** 6  modulo  23  =  9
Enter your Secret key 18
11 ** 18  modulo  23  =  16
So the shared key will be :
9 ** 18  modulo  23  =  4
16 ** 6  modulo  23  =  4
Key exchange successful. Shared key is: 4
```

If we change p and m, it will be :

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p 17
Enter m 8
Enter your Secret key 3
8 ** 3  modulo  17  =  2
Enter your Secret key 16
8 ** 16  modulo  17  =  1
So the shared key will be :
2 ** 16  modulo  17  =  1
1 ** 3  modulo  17  =  1
Key exchange successful. Shared key is: 1
```

and :

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p 17
Enter m 8
Enter your Secret key 7
8 ** 7  modulo  17  =  15
Enter your Secret key 15
8 ** 15  modulo  17  =  15
So the shared key will be :
15 ** 15  modulo  17  =  8
15 ** 7  modulo  17  =  8
Key exchange successful. Shared key is: 8
```

Even if we try with large numbers as secret keys :

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p 37
Enter m 13
Enter your Secret key 789
13 ** 789  modulo  37  =  8
Enter your Secret key 53
13 ** 53  modulo  37  =  17
So the shared key will be :
8 ** 53  modulo  37  =  23
17 ** 789  modulo  37  =  23
Key exchange successful. Shared key is: 23
```

Or even larger, at every spot :

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p 2797
Enter m 1357
Enter your Secret key 458
1357 ** 458  modulo  2797  =  1174
Enter your Secret key 9762
1357 ** 9762  modulo  2797  =  757
So the shared key will be :
1174 ** 9762  modulo  2797  =  1613
757 ** 458  modulo  2797  =  1613
Key exchange successful. Shared key is: 1613
```

What about if the 2 secrets are equal ?

```
The parameters are p = 23 and g =  11
Do you want to change the parameters ? 0 : no, 1 : yes 0
Enter your Secret key 5
11 ** 5  modulo  23  =  5
Enter your Secret key 5
11 ** 5  modulo  23  =  5
So the shared key will be :
5 ** 5  modulo  23  =  20
5 ** 5  modulo  23  =  20
Key exchange successful. Shared key is: 20
```

We can see that the algorithm still works and does not always give a shared key equal to the secret keys.
And if the parameters p and g are negative ?

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p -23
Enter m -11
Enter your Secret key 60
-11 ** 60  modulo  -23  =  -5
Enter your Secret key 40
-11 ** 40  modulo  -23  =  -7
So the shared key will be :
-5 ** 40  modulo  -23  =  -17
-7 ** 60  modulo  -23  =  -17
Key exchange successful. Shared key is: -17
```

It still works but the result is negative. If we apply the verification, so that g has to be inferior to p then the program won't let us try that, as it's not included in the Diffie-Hellman key exchange protocol. So it would work but it's not part of the protocol.

And finally, with very small numbers :

```
Do you want to change the parameters ? 0 : no, 1 : yes 1
Enter p 5
Enter m 3
Enter your Secret key 4
3 ** 4  modulo  5  =  1
Enter your Secret key 6
3 ** 6  modulo  5  =  4
So the shared key will be :
1 ** 6  modulo  5  =  1
4 ** 4  modulo  5  =  1
Key exchange successful. Shared key is: 1
```

We can verify this one with simple calculations :

$3^4 = 81$ and $81 \, mod \, 5 = 1$

$3^6 = 729$ and $729 \, mod \, 5 = 4$

$1^6 = 1$ and $1 \, mod \, 5 = 1$

$4^4 = 256$ and $256 \, mod \, 5 = 1$

Description of the code :

This Python script is an interactive program designed to demonstrate a Diffie-Hellman key exchange protocol. It provides a menu-based interface for users to perform actions such as running the program, modifying parameters, or exiting.
At the beginning of the script in the main we got the import of the Diffie_Hellman function and the import of the sqrt from the math library.
After the import we got the variable of the program with default parameters.
The parameter represents the prime number p in the Diffie Hellman protocol.
The message represents the base g of the protocol.
The “menu” parameter is used to register the user's choice in the menu.
The function is_prime is a function used in the main to verify if a number is a prime number or not. This function returns a boolean.

The main program loop is a while loop which is used as the main structure of the program. It uses a menu interface in the terminal for the user to interact with the code.
The menu as 3 option :
**Option 1 : Run the program :** this option calls the Diffie Hellman function with the current parameter and message. It validates the result to determine if the key exchange was successful and it outputs the shared key or prints a failure.
**Option 2 : Change parameters :** this option asks the user for a new parameter and a new message. This option also check if the given parameter is a prime number and if the message is a number is less than the parameter and if the message is divisible by the parameter
**Option 3 : Exit :** this option terminates the program.
**Default case :** handle invalid menu inputs

Inputs

- Prime number parameter and integer message.
- User menu selections (1, 2, or 3).
- New values for parameter and message when modifying parameters.

Outputs:

- Interactive menu display.
- Messages indicating success, failure, or errors during the key exchange.
- Prompts for re-entering valid inputs when necessary.

**Prime Validation (is_prime function):** Ensures the prime nature of parameter before usage.
**Key Exchange Logic:** Uses diffie_hellman to compute the shared key and verifies its validity.

**Parameter Updates:** Dynamically updates parameter and message with user inputs, ensuring compliance with constraints.
**Menu Management:** Provides a structured flow for user interaction.