Experiment No.6

Name- Diffie-Hellman Key Exchange Algorithm

Aim: Write a program to implement Diffie-Hellman Key Exchange Algorithm using python

Objectives:

• To understand the principles of symmetric key cryptography.

• To understand the Diffie-Hellman Key exchange algorithm.

• To understand the possible attacks on Diffie-Hellman.

Outcomes: The learner will be able to apply the cryptosystem to ensure secure key exchange between sender and receiver.

Theory:

Diffie Hellman key exchange is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

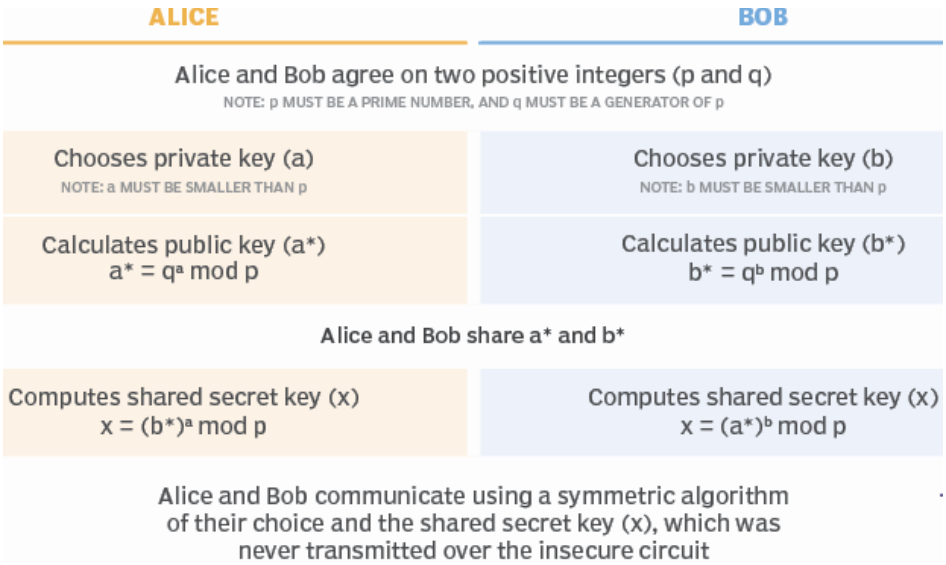
Principles of symmetric Cryptography:

There are two principles for secure use of conventional encryption which are as follows−

It requires a strong encryption algorithm. At a minimum, it is an algorithm to be such that an opponent who understands the algorithm and has access to one or more cipher texts would be inadequate to decipher the cipher text or consider the key. This requirement is generally defined in a stronger form. The opponent would be inadequate to decrypt cipher text or find the key even if it is in possession of multiple cipher texts together with the plaintext that make each cipher text.

Sender and receiver should have obtained copies of the secret key in a secure fashion and should maintain the key secure. If someone can find the key and understand the algorithm, all communication utilizing this key is readable. This characteristic of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm uses not be kept secret means that manufacturers can and have developed inexpensive chip implementations of data encryption algorithms. These chips are broadly available and incorporated into multiple products. With the use of symmetric encryption, the principal security issues are supporting the secrecy of the key. For this reason, the key is sent to the receiver through an independent secure channel. Moreover, a trusted third party can make the key and assign this to both source and destination.

Diffie Hellman Algorithm:



Possible Attacks on Diffie-Hellman

There are several possible attacks on the Diffie-Hellman key exchange protocol, depending on the specific parameters and implementation. Here are some common attacks:

1. Man-in-the-Middle (MitM) Attack: An attacker intercepts the communication between two parties and establishes separate key exchanges with each party. This allows the attacker to decrypt and possibly modify the messages exchanged between the two legitimate parties.

2. Small Key Space Attack: If the parameters used for Diffie-Hellman have a small key space (i.e., a small prime modulus or generator), an attacker can perform a brute-force search to find the shared secret key more easily.

3. Precomputation Attack: An attacker can precompute values for commonly used primes and generators to speed up the key derivation process, making it easier to break the shared secret key when the same parameters are reused.

4. Timing Attacks: By analyzing the timing of cryptographic operations, an attacker may gain information about the private keys involved, potentially leading to the compromise of the shared secret.

5. Invalid Curve Attacks (ECDH): When using Elliptic Curve Diffie-Hellman (ECDH), an attacker might exploit weaknesses in the curve parameters, such as a poorly chosen curve, to compute the shared secret more efficiently.

To defend against these attacks, it's crucial to use strong, well-established parameters, ensure secure key exchange protocols, and implement additional security measures like message authentication codes (MACs) and digital signatures when needed. Regularly updating cryptographic libraries and staying informed about the latest developments in cryptography is also important to mitigate potential vulnerabilities.

Python code:

```
# Diffie-Hellman Code
P = int(input("Enter P: "))
G = int(input("Enter Q: "))
# Private Keys
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
   input("Enter The Private Key Of User 2 : "))
while 1:
   if x1 >= P or x2 >= P:
      print(f"Private Key Of Both The Users Should Be Less Than {P}!")
      continue
   break
# Calculate Public Keys
y1, y2 = pow(G, x1) % P, pow(G, x2) % P
# Generate Secret Keys
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P
print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")
if k1 == k2:
   print("Keys Have Been Exchanged Successfully")
else:
   print("Keys Have Not Been Exchanged Successfully")
```

Output:

```
Enter P: 687
Enter Q: 71
Enter The Private Key Of User 1 : 45
Enter The Private Key Of User 2 : 78

Secret Key For User 1 Is 394
Secret Key For User 2 Is 394

Keys Have Been Exchanged Successfully
```

Conclusion: The Diffie-Hellman key exchange algorithm is used to make a channel to share secret keys between sender and receiver.