CNS ASSIGNMENT 5 - IMPLEMENTATION OF DIFFIE-HELLMAN ALGORITHM

MIS - 112003151

NAME - ADWAIT VIPRA

The Diffie-Hellman key exchange algorithm is a public-key cryptography protocol that allows two parties to establish a shared secret key over an insecure communication channel. Here's a brief overview of the Diffie-Hellman algorithm:

1. **Key Generation:**

   - Both parties, let's call them Alice and Bob, agree on two public parameters: a large prime number (p) and a primitive root modulo (p), often denoted as (g).
   - Each party independently generates a private key: (a) for Alice and (b) for Bob. These keys are kept secret.
2. **Public Key Calculation:**

   - Using the agreed-upon (p) and (g), each party calculates a public key:
     - Alice computes (A = g^a \mod p)
     - Bob computes (B = g^b \mod p)
3. **Key Exchange:**

   - Alice and Bob exchange their public keys ((A) and (B)) over the insecure channel.
4. **Shared Secret Calculation:**

   - Alice, with her private key (a) and Bob's public key (B), computes the shared secret (S = B^a \mod p).
   - Bob, with his private key (b) and Alice's public key (A), computes the same shared secret (S = A^b \mod p).
5. **Result:**

   - Both Alice and Bob now have a shared secret key (S) that can be used for secure communication.
   - The strength of the Diffie-Hellman algorithm relies on the difficulty of the discrete logarithm problem, making it computationally infeasible for an eavesdropper to determine the shared secret even if the public keys are intercepted.

The Diffie-Hellman key exchange is a foundational concept in modern cryptography and is widely used in various protocols, including secure socket layer (SSL) for securing web communication. It enables secure key establishment between parties without the need for pre-shared keys, allowing for secure communication over untrusted networks.

In [ ]:
```
#!/bin/python

'''
p is large prime
q is primitve root of p

a:
xa: private key
ya: public key (q^xa)mod(p)

b:
xb: private key
yb: public key (q^xb)mod(p)

ka = (yb^xa)mod(p)
kb = (ya^xb)mod(p)
'''
```

In [5]:
```
def isvalid (p,xa):
    if xa < p:
        return True
    return False

def main ():
    p = int(input("p ="))
    q = int(input("q ="))

    xa = int(input("a ="))
    xb = int(input("b ="))

    if isvalid (p, xa) and isvalid (p, xb):
        ya = pow(q, xa) % p
        yb = pow(q, xb) % p

        print("ya =", ya)
        print("yb =", yb)

        ka = pow(yb, xa) % p
        kb = pow(ya, xb) % p

        print ("ka =", ka)
        print ("kb =", kb)
    else:
        print("Invalid private keys!\n")

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

```
p =23
q =5
a =6
b =15
ya = 8
yb = 19
ka = 2
kb = 2
```