STUDY OF ASYMMETRIC KEY (DH & RSA)

Roll No.: 18001

AIM:

To study and implement Asymmetric Key Cryptography using **Diffie–Hellman (DH)** and **RSA** algorithms in Python.

DIFFIE–HELLMAN (DH) ALGORITHM

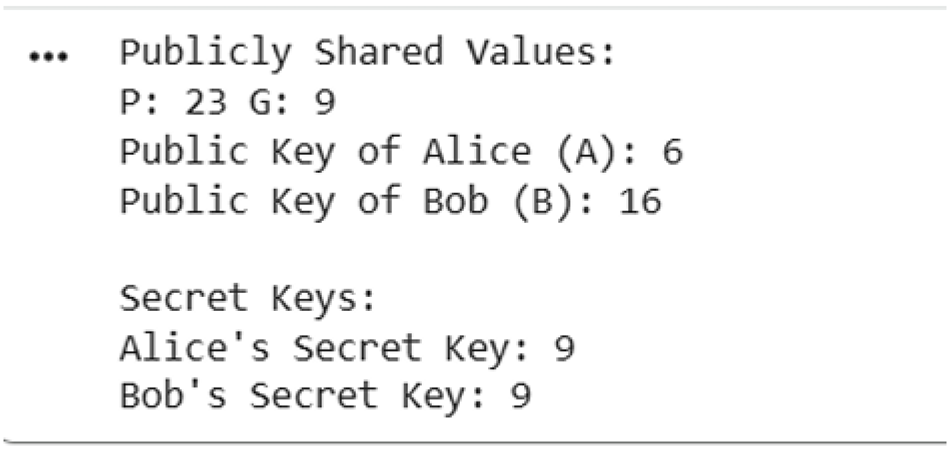
Algorithm Steps

1. Select a large prime number P and a primitive root G.
2. Each user selects a private key (random number).
   - Alice selects **a**
   - Bob selects **b**
3. Compute the public keys:
   - $A = G^a \bmod P$
   - $B = G^b \bmod P$
4. Exchange the public keys between Alice and Bob.
5. Compute the shared secret key:
   - $Key1 = B^a \bmod P$ (Alice computes)
   - $Key2= A^b \bmod P$ (Bob computes)
6. Both keys must be identical.

PROGRAM – DIFFIE–HELLMAN

```
P = 23  # prime number
G = 9   # primitive root

a = 4   # Alice's private key
b = 3   # Bob's private key

A = (G ** a) % P   # Alice's public key
B = (G ** b) % P   # Bob's public key

key1 = (B ** a) % P  # Alice's shared key
key2 = (A ** b) % P  # Bob's shared key

print("Publicly Shared Values:")
print("P:", P, "G:", G)
print("Public Key of Alice (A):", A)
print("Public Key of Bob (B):", B)

print("\nSecret Keys:")
print("Alice's Secret Key:", key1)
print("Bob's Secret Key:", key2)
```

OUTPUT :-



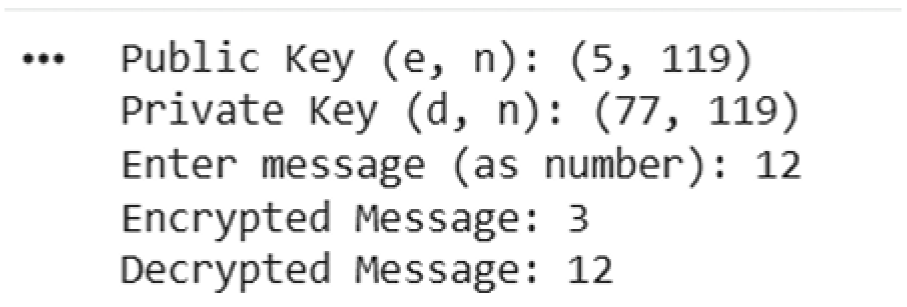
```
Publicly Shared Values:
P: 23 G: 9
Public Key of Alice (A): 6
Public Key of Bob (B): 16

Secret Keys:
Alice's Secret Key: 9
Bob's Secret Key: 9
```

ALGORITHM OF RSA :- 1. Choose two prime numbers p and q.
2. Compute n = p × q.
3. Compute Euler's totient function φ(n) = (p − 1) × (q − 1).
4. Choose a public key e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1.
5. Compute private key d such that (d × e) mod φ(n) = 1.
6. For encryption: C = (M^e) mod n.
7. For decryption: M = (C^d) mod n.

PROGRAM – RSA

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

p = 7
q = 17
n = p * q
phi = (p - 1) * (q - 1)

e = 5
while gcd(e, phi) != 1:
    e += 1

d = pow(e, -1, phi)

print("Public Key (e, n):", (e, n))
print("Private Key (d, n):", (d, n))

msg = int(input("Enter message (as number): "))
cipher = pow(msg, e, n)
print("Encrypted Message:", cipher)

decrypted = pow(cipher, d, n)
print("Decrypted Message:", decrypted)
```

OUTPUT :-



```
Public Key (e, n): (5, 119)
Private Key (d, n): (77, 119)
Enter message (as number): 12
Encrypted Message: 3
Decrypted Message: 12
```

CONCLUSION :- Asymmetric cryptography uses two different keys for secure communication. In this practical DH was used to generate secret keys, and RSA was used for message encryption and decryption, demonstrating secure data exchange.