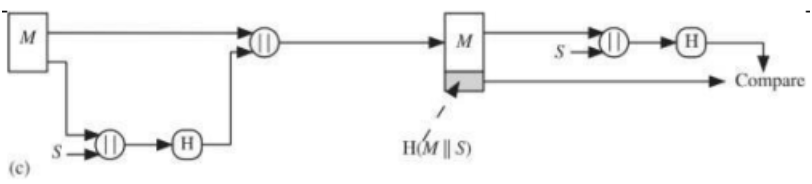
**PRACTICAL: 7**

**AIM:**

Refer to the attached figure here. Bob is preparing to send message to Alice. Bob and Alice both secretly computes the code(s) without sharing on any communication channel. Suggest key exchange algorithm to Bob and Alice for securely exchange information without sharing actual key. Once they form secret code, Bob applies SHA256 hash algorithm on original message (M) plus code (s) and send hash of original message and code (M||s) to Alice. Alice will receive bundle of H(M||s) and first append code (s) with received message (M) and produce hash of the message (H) that compare with H(M||s) to make sure that message is not altered by any attackers.



The task to perform:

1. Use some key exchange algorithm to calculate the value of s (secret code) which must be unique at the sender and receiver side.

2. Implementation can be done using any programming language such as Java programming or python programming.

3. Apply SHA256 on t h e message and secret code and display it on the output screen. Verify the hash value at the receiver end.

**THEORY:**

Key exchange algorithms enable two parties to establish a shared secret without directly transmitting it over insecure channels. **Diffie-Hellman (DH)** and **Elliptic Curve Diffie-Hellman (ECDH)** are widely used for secure key exchange, ensuring that both sender and receiver derive the same secret key independently. Once the secret key is established, cryptographic hash functions like **SHA256** are applied to ensure data integrity. Hashing combines the message with the secret code, generating a unique hash. At the receiver’s end, recalculating and verifying the hash ensures message authenticity, preventing unauthorized modifications and enhancing secure communication.

**CODE/COMMAND:**

import hashlib

import random

# 1. Diffie-Hellman Key Exchange

def generate\_keys(p, g):

    private\_key = random.randint(1, p - 1)  # Private key

    public\_key = pow(g, private\_key, p)    # Public key

    return private\_key, public\_key

# 2. Compute Shared Secret

def compute\_shared\_secret(private\_key, received\_public, p):

    return pow(received\_public, private\_key, p)

# 3. Hashing the message with secret code

def sha256\_hash(message, secret):

    combined = message + str(secret)

    return hashlib.sha256(combined.encode()).hexdigest()

# Public parameters (agreed upon)

p = 23  # Large prime number

g = 5   # Primitive root

# Bob and Alice generate their keys

bob\_private, bob\_public = generate\_keys(p, g)

alice\_private, alice\_public = generate\_keys(p, g)

# Exchange public keys and compute the shared secret

bob\_secret = compute\_shared\_secret(bob\_private, alice\_public, p)

alice\_secret = compute\_shared\_secret(alice\_private, bob\_public, p)

# Both secrets should be the same

assert bob\_secret == alice\_secret, "Key exchange failed!"

secret\_code = bob\_secret  # This will be used for hashing

# Message to be sent

message = "Hello Alice, this is Bob!"

print("Bob sends message: ", message)

# Bob hashes the message with the secret code

hashed\_message = sha256\_hash(message, secret\_code)

print("Bob sends hashed message: ", hashed\_message)

# Alice receives the message and verifies

received\_hash = sha256\_hash(message, secret\_code)

print("Alice verifies: ", received\_hash)

# Checking integrity

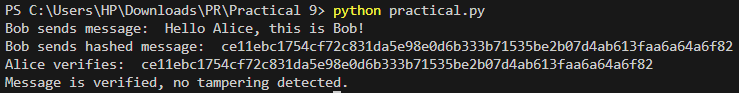
if received\_hash == hashed\_message:

    print("Message is verified, no tampering detected.")

else:

    print("Message integrity check failed!")

**OUTPUT:**



**LATEST APPLICATIONS:**

* **Secure Messaging (Signal, WhatsApp)** – Uses key exchange algorithms like ECDH for end-to-end encryption.
* **TLS/SSL in Web Security** – Diffie-Hellman key exchange secures HTTPS connections for safe online transactions.
* **Cryptocurrency Wallets** – Blockchain systems use ECDH for secure key sharing in multi-signature wallets.

**LEARNING OUTCOMES:**

* Learn how key exchange algorithms establish a shared secret without direct transmission.
* Understand the role of SHA256 in ensuring message integrity and authentication.
* Implement and verify cryptographic techniques for secure communication.