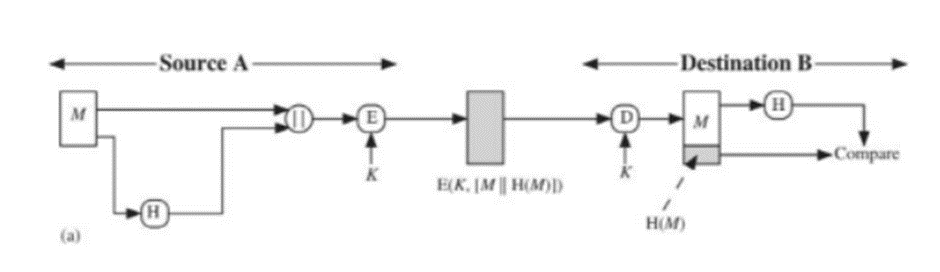
**PRACTICAL 6**

**Aim:** Refer to the figure (a) attached here. Bob (Source A) is preparing to send message to Alice (Destination B). Bob applies SHA256 hash algorithm on prepared message and append with original message (M) which is further encrypted by single secret key. Alice will receive bundle of encrypted H(M) and original message (M). Alice will first apply single secret key to decrypt the entire bundle and collect H(M) and original message (M). Furthermore, Alice will apply the same algorithm SHA256 which was used by Bob and produce hash of received message (H). Lastly, Alice will verify the computed hash with received H(M) to make sure message is not altered by any attackers.



Task to perform:

1) Use any Symmetric key/Asymmetric key algorithm to implement encryption function and decryption.

2) Implementation can be done using any programming language such as c,

c++, java, python, c#, javascript, etc.

3) For SHA256 hashing, you may use library compatible as per your programming language.

Discuss the issues causes with this scenario. What happened if we encrypt the generated hash?

**Hardware Requirement:** Computer/Laptop.

**Software Requirement:** Any Python IDE.

**Theory:**

The SHA-256 algorithm is a variant of SHA-2 (Secure Hash Algorithm 2), a successor to SHA-1 developed by the National Security Agency in 2001. The SHA-256 hash function is a patented cryptographic hash function that generates a 256-bit result.

What exactly is hashing? Encryption converts data into a secure format that cannot be read unless the recipient possesses a key. The data can be any size in encrypted form, and it's typically just as long as it is unencrypted. In hashing, on the other hand, data of any size is mapped to data of a specific size. A 512-bit string of data, for example, would be turned into a 256-bit string using SHA-256 hashing.

**CODE:**

**P6.py**

import hashlib

from cryptography.fernet import Fernet

def hash(msg: str):

    return hashlib.sha256(msg.encode('utf-8')).hexdigest()

def prepare\_msg\_string(msg: str):

    prepared\_msg\_to\_encrypt = str(hash(msg))+ "," + str(msg)

    print("Prpared Msg To Encrypt: ", prepared\_msg\_to\_encrypt, "\n")

    return prepared\_msg\_to\_encrypt

def encrypt\_msg(msg: str, key):

    return key.encrypt(bytes(msg, 'utf-8'))

def decrypt\_msg(msg: str, key):

    return key.decrypt(msg)

def get\_msg\_to\_be\_sent(msg: str, encryption\_key):

    hashed\_msg = hash(msg)

    msg\_string = prepare\_msg\_string(msg)

    encrypted\_msg\_string = encrypt\_msg(msg\_string, encryption\_key)

    return encrypted\_msg\_string

def verify\_msg(msg: str, key):

    msg = decrypt\_msg(msg, key).decode()

    print("Decrypted Msg: ", msg)

    msg = msg.split(",")

    hash\_of\_original\_msg = hashlib.sha256(msg[1].encode('utf-8')).hexdigest()

    if(msg[0] == hash\_of\_original\_msg):

        print("Original Msg: ", msg[1])

        print("Msg Is Non-Compromised")

    else:

        print("Hash Varyfication Failed Msg Is Compromised")

def generate\_key():

    key = Fernet.generate\_key()

    key = Fernet(key)

    return key

def main():

    key=generate\_key()

    print(key)

    print("Enter Msg To Be Sent : ")

    msg=input()

    print("Msg To Be Sent : ", msg)

    msg\_sended = get\_msg\_to\_be\_sent(msg, key).decode()

    print("Msg Sended: ", msg\_sended,"\n\n")

    msg\_received = msg\_sended

    print("Msg Recieved: ", msg\_received, "\n\n")

    verify\_msg(msg\_received.encode(), key)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Server.py**

import P6

import socket,pickle

def server\_program():

    # get the hostname

    host = socket.gethostname()

    port = 5000  # initiate port no above 1024

    server\_socket = socket.socket()  # get instance

    # look closely. The bind() function takes tuple as argument

    server\_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously

    server\_socket.listen(2)

    conn, address = server\_socket.accept()  # accept new connection

    print("Connection from: " + str(address))

    # reckey=conn.recv(4096).decode()

    # key=pickle.loads(reckey)

    msg=conn.recv(1024).decode()

    print("Msg Recieved: ", msg,"\n\n")

    print("Sending Recieved Messege to verify \n\n")

    conn.send(msg.encode())

if \_\_name\_\_ == '\_\_main\_\_':

    server\_program()

**Client.py**

from copyreg import pickle

import P6

import socket,pickle

def client\_program():

    host = socket.gethostname()  # as both code is running on same pc

    port = 5000  # socket server port number

    client\_socket = socket.socket()  # instantiate

    client\_socket.connect((host, port))  # connect to the server

    key=P6.generate\_key()

    # sendkey=pickle.dump(key)

    # client\_socket.send(sendkey.encode())

    print("Enter Msg To Be Sent : ")

    msg=input()

    print("Msg To Be Sent : ", msg)

    msg\_sended = P6.get\_msg\_to\_be\_sent(msg, key).decode()

    print("Msg Sended: ", msg\_sended,"\n\n")

    client\_socket.send(msg\_sended.encode())

    print("\n\n")

    msg\_verify=client\_socket.recv(1024).decode()

    P6.verify\_msg(msg\_verify.encode(), key)

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**OUTPUT:**

