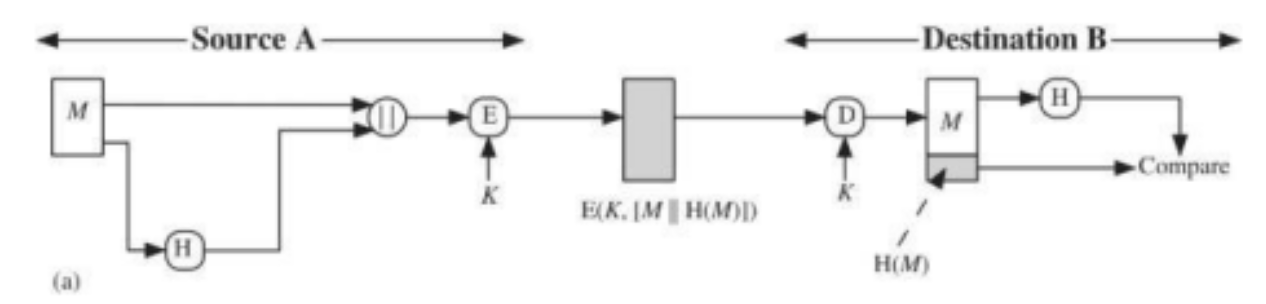
**PRACTICAL: 6**

**AIM:**

Refer to the figure (a) attached here. Bob (Source A) is preparing to send a message to Alice (Destination B). Bob applies the SHA256 hash algorithm on the prepared message and appends it with original message (M) which is further encrypted by a single secret key. Alice will receive a bundle of encrypted H(M) and original messages (M). Alice will first apply a single secret key to decrypt the entire bundle and collect H(M) and the original message (M). Furthermore, Alice will apply the same algorithm SHA256 which was used by Bob, and produce a hash of the received message (H). Lastly, Alice will verify the computed hash with the received H(M) to make sure the 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 Java programming or python programming.

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?

**THEORY:**

Cryptographic security relies on hashing and encryption to ensure data integrity, confidentiality, and authenticity. **SHA256** is a cryptographic hash function that converts input data into a fixed-size 256-bit hash, ensuring data integrity since even a small change in input results in a completely different hash (avalanche effect).

**Encryption**, whether symmetric (e.g., AES) or asymmetric (e.g., RSA), transforms plaintext into ciphertext, ensuring confidentiality. Hashes are typically not encrypted, as they are used for verification, not secrecy. However, encrypting a hash can provide additional security in authentication protocols but does not prevent unauthorized data modification.

**CODE/COMMAND:**

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

import hashlib

import os

AES\_KEY = os.urandom(16)

def sha256\_hash(message):

    return hashlib.sha256(message.encode()).digest()

def encrypt\_message(message, key):

    hashed\_msg = sha256\_hash(message)

    # Concatenate message and hash

    enc\_msg = message.encode() + hashed\_msg

    # Initialize AES cipher in CBC mode

    cipher = AES.new(key, AES.MODE\_CBC, iv=os.urandom(16))

    iv = cipher.iv

    # Encrypt with padding

    encrypted\_message = cipher.encrypt(pad(enc\_msg, AES.block\_size))

    return iv + encrypted\_message  # Prepend IV for decryption

def decrypt\_message(encrypted\_message, key):

    """Decrypt message and extract hash using AES CBC mode."""

    iv = encrypted\_message[:16]  # Extract IV

    encrypted\_message = encrypted\_message[16:]  # Extract encrypted data

    cipher = AES.new(key, AES.MODE\_CBC, iv)

    decrypted\_data = unpad(cipher.decrypt(encrypted\_message), AES.block\_size)

    # Extract original message and hash

    message = decrypted\_data[:-32].decode()

    received\_hash = decrypted\_data[-32:]

    return message, received\_hash

def verify\_message(message, received\_hash):

    """Verify if the received hash matches the computed hash."""

    computed\_hash = sha256\_hash(message)

    return computed\_hash == received\_hash

# Example Usage

original\_message = "Hello Alice! This is Bob."

print(f"Original Message: {original\_message}")

# Bob encrypts the message

encrypted\_data = encrypt\_message(original\_message, AES\_KEY)

print(f"Encrypted Data: {encrypted\_data.hex()}")

# Alice decrypts the message

received\_message, received\_hash = decrypt\_message(encrypted\_data, AES\_KEY)

print(f"Decrypted Message: {received\_message}")

# Alice verifies integrity

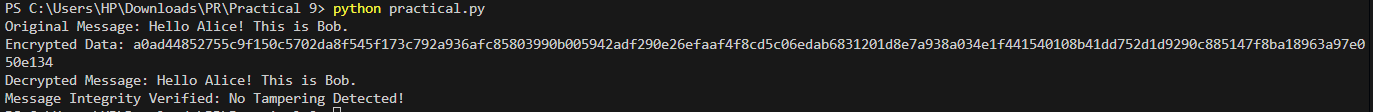
if verify\_message(received\_message, received\_hash):

    print("Message Integrity Verified: No Tampering Detected!")

else:

    print("Warning: Message Integrity Compromised!")

**OUTPUT:**



**LATEST APPLICATIONS:**

* **Blockchain Technology** – SHA256 is widely used in Bitcoin and other cryptocurrencies for transaction integrity and proof-of-work consensus mechanisms.
* **Secure Messaging Apps** – End-to-end encrypted platforms like WhatsApp and Signal use AES and RSA encryption with hashing to ensure message authenticity and confidentiality.
* **Digital Signatures & Authentication** – Modern authentication systems use hashing and encryption for password hashing (e.g., Argon2, bcrypt) and secure digital signatures (e.g., ECDSA).

**LEARNING OUTCOMES:**

* Understand the role of cryptographic hashing (SHA256) in ensuring data integrity and security.
* Implement symmetric/asymmetric encryption techniques to protect sensitive information.
* Analyze potential security risks and the impact of encrypting hash values in communication systems.