**Name - Gaurang A Raorane Roll No - 49**

**Class - D15A Batch - C**

**EXPERIMENT 7**

**Aim:-** Write a Program to implement Cryptographic Hash Functions and Applications (HMAC): To Understand the need, design and applications of collision resistant hash functions.

**Theory:-**

**HMAC (Hash-based Message Authentication Code):**

* + HMAC is a widely used technique in cryptography and network security.
  + It involves the use of a cryptographic hash function and a secret key to ensure the integrity and authenticity of a message or data.
  + HMAC creates a unique and fixed-size hash value for a given message, which can be used for verification.
* Integrity and Authenticity:
  + Data integrity ensures that the data has not been altered or corrupted during transmission or storage.
  + Authenticity verifies that the data indeed originates from the claimed sender and has not been tampered with by unauthorized parties.
* SHA-256 Hash Function:
  + SHA-256 is a specific cryptographic hash function used in this program.
  + It generates a fixed-size hash value (256 bits) from an input message, making it difficult for attackers to reverse-engineer the original data.
* Secret Key:
  + A secret key is a critical component of HMAC.
  + It is known only to the sender and the receiver and is used to compute and verify the HMAC.
  + The secrecy of the key is vital for the security of the system.
* Code Implementation:
  + The program demonstrates how HMAC can be computed in Java using the SHA-256 hash function and a secret key.
  + It initializes the HMAC algorithm with the secret key and computes the HMAC hash for a given message.
  + The result is displayed as a Base64 string, which is a common encoding format for hash values.
* Security Considerations:
  + HMAC is used to enhance data security, but the effectiveness of HMAC relies on the strength and secrecy of the secret key.
  + In real-world applications, secure key management practices are essential to protect against unauthorized access.

**Input:-**

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.security.NoSuchAlgorithmException;

import java.security.InvalidKeyException;

import java.util.Base64;

public class SimpleHMACExample {

public static void main(String[] args) {

// Define a secret key (keep this secret)

String secretKey = "mySecretKey";

String message = "Hello, HMAC!";

try {

// Create a SHA-256 HMAC instance

Mac hmac = Mac.getInstance("HmacSHA256");

SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey.getBytes(), "HmacSHA256");

hmac.init(secretKeySpec);

// Compute the HMAC hash

byte[] hmacBytes = hmac.doFinal(message.getBytes());

// Encode the HMAC hash as a Base64 string

String encodedHMAC = Base64.getEncoder().encodeToString(hmacBytes);

System.out.println("Message: " + message);

System.out.println("HMAC (SHA-256): " + encodedHMAC);

} catch (NoSuchAlgorithmException | InvalidKeyException e) {

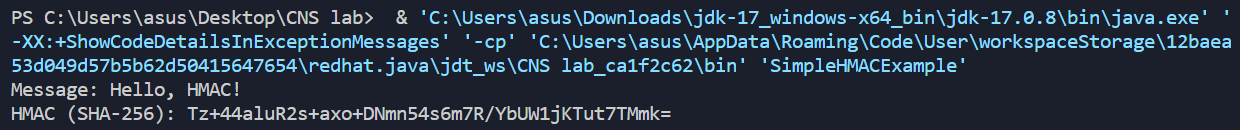
e.printStackTrace();

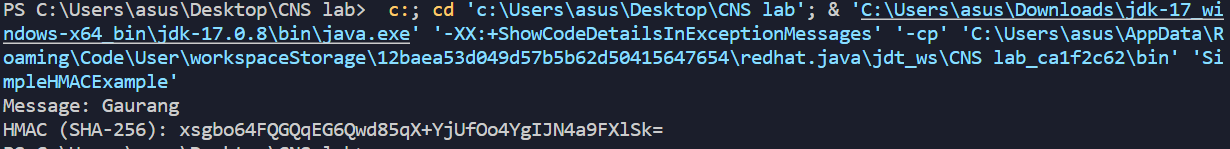
}

}

}

**Output:-**

****

****

**Conclusion:-**

Implemented HMAC cryptographic hash function successfully.