**Assignment No 11**

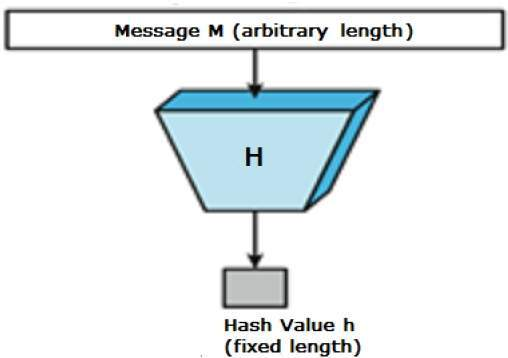
**Title :** To implement Cryptographic Hash Functions

**PRN :** 2017BTECS00215

**Theory :**

Hash functions are extremely useful and appear in almost all information security applications.

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.

Values returned by a hash function are called message digest or simply hash values. The following picture illustrated hash function

***Message Digest (MD):***

MD5 was the most popular and widely used hash function for quite some years. The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as an Internet Standard RFC 1321. It is a 128-bit hash function. MD5 digests have been widely used in the software world to provide assurance about the integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it. In 2004, collisions were found in MD5. An analytical attack was reported to be successful in only an hour by using a computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

***Secure Hash Algorithm (SHA) :***

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from the same family, there are structurally different. The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0. SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful. SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on the number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function. Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs. In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

**Program :**

import java.util.Scanner;

import java.math.BigInteger; // For output conversion

import java.security.MessageDigest; // For cryptographic hash functions

import java.security.NoSuchAlgorithmException; // Handle exception

class Hash\_Test

{

public static String hash(String msg, int opt)

{

try

{

MessageDigest md;

// Get the hash instance

switch(opt)

{

case 1:

md=MessageDigest.getInstance("MD2");

break;

case 2:

md=MessageDigest.getInstance("MD5");

break;

case 3:

md=MessageDigest.getInstance("SHA-1");

break;

case 4:

md=MessageDigest.getInstance("SHA-256");

break;

case 5:

md=MessageDigest.getInstance("SHA-512");

break;

default:

md=MessageDigest.getInstance("MD2"); // default

break;

}

// Calculate hash value

byte[] hashValue=md.digest(msg.getBytes());

// Convert byte to number

BigInteger num=new BigInteger(1,hashValue);

// Convert to hex value

String messageDigest=num.toString(16);

while(messageDigest.length()<32)

{

messageDigest="0"+messageDigest;

}

return messageDigest;

}

catch (NoSuchAlgorithmException e)

{

throw new RuntimeException(e);

}

}

public static void main(String args[]) throws NoSuchAlgorithmException

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter message : ");

String msg=sc.nextLine();

System.out.print("Choose algorithm\n1. MD2\n2. MD5\n3. SHA-128\n4. SHA-256\n5. SHA-512\nOption : ");

int opt=sc.nextInt();

if(opt<1 && opt>5)

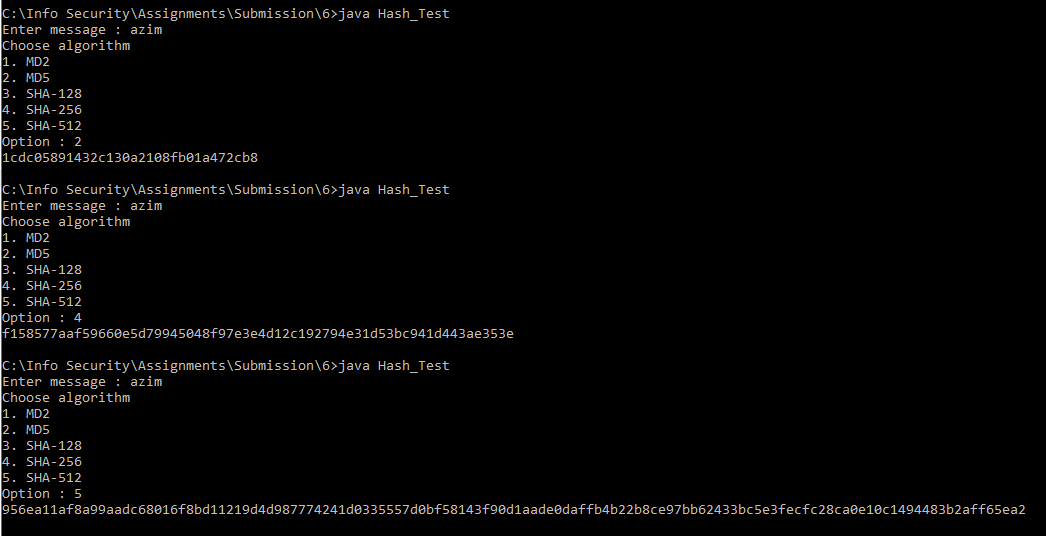
System.out.println("Invalid option!");

else

System.out.println(hash(msg,opt));

}

}

**Output:**

**Conclusion :**

In this practical, we studied different cryptographic hash functions and learnt how to use/implement them for their realization.