**LAB 6**

**OBJECTIVE**

To implement cryptographic hash functions

**THEORY**

A cryptographic hash function (CHF) is a hash function that is suitable for use in cryptography. It is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit string of a fixed size (the "hash value", "hash", or "message digest") and is a one-way function, that is, a function which is practically infeasible to invert. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.

The ideal cryptographic hash function has the following main properties:

* It is deterministic, meaning that the same message always results in the same hash.
* It is quick to compute the hash value for any given message.
* It is infeasible to generate a message that yields a given hash value.
* It is infeasible to find two different messages with the same hash value.
* A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value (avalanche effect).

Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption.

**CODE**

# Python program to implement different hashing algorithms

import hashlib

message = "Tirtha Raj Poudel"

# MD-5

md5 = hashlib.md5(message.encode())

print(f"MD-5 hash message digest: \n{md5.hexdigest()}")

# SHA-1

sha1 = hashlib.sha1(message.encode())

print(f"\nSHA-1 hash message digest: \n{sha1.hexdigest()}")

# SHA-224

sha224 = hashlib.sha224(message.encode())

print(f"\nSHA-224 hash message digest: \n{sha224.hexdigest()}")

# SHA-256

sha256 = hashlib.sha256(message.encode())

print(f"\nSHA-256 hash message digest: \n{sha256.hexdigest()}")

# SHA-384

sha384 = hashlib.sha384(message.encode())

print(f"\nSHA-384 hash message digest: \n{sha384.hexdigest()}")

# SHA-512

sha512 = hashlib.sha512(message.encode())

print(f"\nSHA-512 hash message digest: \n{sha512.hexdigest()}")

**OUTPUT**

MD-5 hash message digest:

73b43299988c5a8c6b9ace17a5899551

SHA-1 hash message digest:

005a12dee677fe410b73c6b00803f6f063add451

SHA-224 hash message digest:

9f355a709635a7c89a858bdbd75e917dde3b1f21aec13de52166c504

SHA-256 hash message digest:

c7aff9951e9bcd7f75ec6c0612d46abc534d3f6f78c4018367ff6da27e4ff465

SHA-384 hash message digest:

9080369e82432c94c6d0d7681cf8a9845bdac59b9d51cd2a0f89957d81893b2ece7863954c3f3a71ff20eec052b747e5

SHA-512 hash message digest:

ebf5c60d0b5b14b90cd68b2b4349f4cef1c5c7314b0f8d1a5cb4ac525233ca8371ac08a844c5d017a081fa08f3b1586a951c6ce97d381b04ad5b700bf046da76

**CONCLUSION**

In this lab, we got familiar with different cryptographic hashing algorithms and implemented them using python programming language and observed different message digests for different message inputs.