**Experiment no.5**

**Aim:** To study and understand hashing algorithm.

**Learning Objective:** Student should be able to understand about hashing function and its algorithm like MD5,SHA etc.

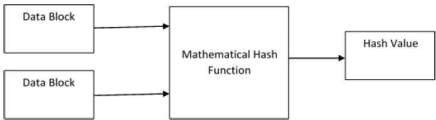
**Tools:** C++/Java/Python

**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.

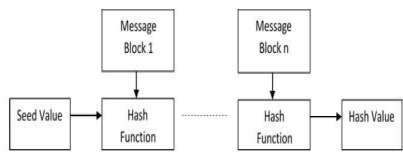
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function −



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration −

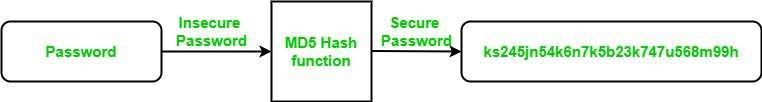


Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an **avalanche** effect of hashing.

**MD5** is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes. MD5 algorithm stands for the **message-digest algorithm**. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always **128 bits. MD5** wasdeveloped in 1991 by **Ronald Rivest.**

**Use Of MD5 Algorithm:**

* It is used for file authentication.
* In a web application, it is used for security purposes. e.g. Secure password of users etc.  Using this algorithm, We can store our password in 128 bits format.

 *MD5 Algorithm*

**Implementation :**

Code :

import math

import hashlib

rotate\_by = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,

5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20,

4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,

6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]

constants = [int(abs(math.sin(i+1)) \* 4294967296) & 0xFFFFFFFF for i in range(64)]

def pad(msg):

msg\_len\_in\_bits = (8\*len(msg)) & 0xffffffffffffffff

msg.append(0x80)

while len(msg)%64 != 56:

msg.append(0)

msg += msg\_len\_in\_bits.to\_bytes(8, byteorder='little')

return msg

init\_MDBuffer = [0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476]

def leftRotate(x, amount):

x &= 0xFFFFFFFF

return (x << amount | x >> (32-amount)) & 0xFFFFFFFF

def processMessage(msg):

init\_temp = init\_MDBuffer[:]

for offset in range(0, len(msg), 64):

A, B, C, D = init\_temp

block = msg[offset : offset+64]

for i in range(64):

if i < 16:

func = lambda b, c, d: (b & c) | (~b & d)

index\_func = lambda i: i

elif i >= 16 and i < 32:

func = lambda b, c, d: (d & b) | (~d & c)

index\_func = lambda i: (5\*i + 1)%16

elif i >= 32 and i < 48:

func = lambda b, c, d: b ^ c ^ d

index\_func = lambda i: (3\*i + 5)%16

elif i >= 48 and i < 64:

func = lambda b, c, d: c ^ (b | ~d)

index\_func = lambda i: (7\*i)%16

F = func(B, C, D)

G = index\_func(i)

to\_rotate = A + F + constants[i] + int.from\_bytes(block[4\*G : 4\*G + 4], byteorder='little')

newB = (B + leftRotate(to\_rotate, rotate\_by[i])) & 0xFFFFFFFF

A, B, C, D = D, newB, B, C

for i, val in enumerate([A, B, C, D]):

init\_temp[i] += val

init\_temp[i] &= 0xFFFFFFFF

return sum(buffer\_content<<(32\*i) for i, buffer\_content in enumerate(init\_temp))

def MD\_to\_hex(digest):

raw = digest.to\_bytes(16, byteorder='little')

return '{:032x}'.format(int.from\_bytes(raw, byteorder='big'))

def md5(msg):

msg = bytearray(msg, 'ascii')

msg = pad(msg)

processed\_msg = processMessage(msg)

message\_hash = MD\_to\_hex(processed\_msg)

print("Hash Value: ", message\_hash)

def hash\_value(msg):

hashvalue = hashlib.md5(msg.encode()).hexdigest()

print("Hash value using hashlib: ", hashvalue)

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

print ("Enter the message to be hashed: ")

message = input()

md5(message)

hash\_value(message)

**Output:**

Enter the message to be hashed: Thakur College

Hash Value: 4082e278a88b6458bab5c705b0b07b7e

Hash value using hashlib: 3141df0027caeb3659c237145ed6b404

**Result and Discussion :** In this experiment we successfully understood the concept of Hashing function algorithm and implemented the MD5 algorithm using python.

**Learning Outcomes:** The student will be able to

LO1: Understand the Concept of Hashing Functions

LO2: Understand the Steps for implementing the hashing function algorithm.

**Course Outcomes:** Upon completion of the course students will be able to study the various network reconnaissance tools & how to use them to gather primary network information.

**Conclusion:** We have implemented hashing algorithms and understoodthe concept of hash value algorithms

**For Faculty Use**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction**  **Parameter s** | **Formative**  **Assessment**  **[40%]** | **Timely completion of Practical**  **[ 40%]** | **Attendance**  **/ Learning**  **Attitude**  **[20%]** |  |
| **Marks**  **Obtained** |  |  |  |