FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING
Department of Computer Engineering

Course , Subject & Experiment Details

Academic Year	2024-25	Estimated Time	02 - Hours
Course & Semester	T.E. (CMPN)- Sem VI	Subject Name & Code	CSS - (CSC602)
Module No.	03 – Mapped to CO- 3	Chapter Title	Cryptographic Hash Functions

Practical No:	6
Title:	Performance Analysis of Hash Algorithms
Date of Performance:	27/03/2025
Date of Submission:	27/04/2025
Roll No:	9913
Name of the Student:	Mark Lopes

Evaluation:

Sr. No	Rubric	Grade
1	On time submission Or completion (2)	
2	Preparedness(2)	
3	Skill (4)	
4	Output (2)	

Signature of the Teacher:

Date:

Title: For varying message sizes, test integrity of message using MD-5, SHA-1, and analyse the performance of the two protocols.

Lab Objective :

This lab provides insight into:

- The working of MD5 and SHA-1 and variations of SHA-1 and analyze the performance of both for varying message sizes.

Reference : "Cryptography and Network Security" B. A. Forouzan
"Cryptography and Network Security" Atul Kahate
www.md5summer.org/download.html

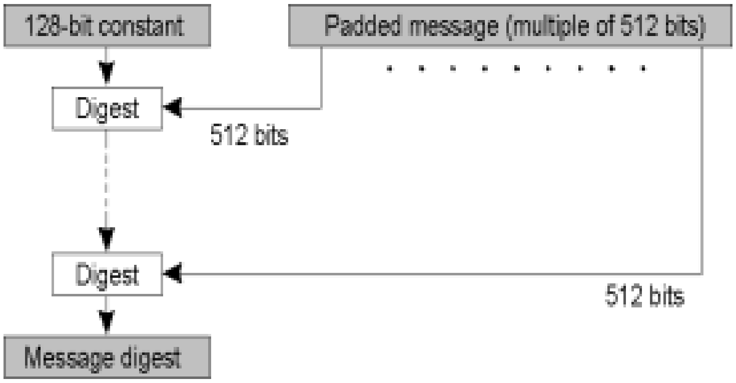
Prerequisite: Java or Python and Knowledge of hashing and Crypt API.

Theory:

Cryptographic hash functions are a very useful tool in cryptography. They are applied in many areas like integrity of messages, storage of passwords securely and protect signatures. The three hash algorithms SHA-1, SHA-512 and MD5 are considered to analyze their performance.

MD5

- Takes as input a message of arbitrary length and produces as output a 128 bit "fingerprint" or "message digest" of the input.
- It is conjectured that it is computationally infeasible to produce two messages having the same message digest.
- Intended where a large file must be "compressed" in a secure manner before being encrypted with a private key under a public-key cryptosystem such as PGP



Input:

Suppose a b-bit message as input, and that we need to find its message

digest. **Algorithm:**

Step 1 – append padding bits:

– The message is padded so that its length is congruent to 448, modulo 512.
- Means extended to just 64 bits of being of 512 bits long.
– A single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits equals 448 modulo 512.

• Step 2 – append length

– A 64 bit binary representation of b is appended to the result of the previous step.
– The resulting message has a length that is an exact multiple of 512 bits.

• Step 3 – Divide the input into 512-bit blocks

Now we divide the input mesg into into blocks , each of length 512 bits.

• Step 4 – Initialize MD Buffer

- A four-word buffer (A,B,C,D) is used to compute the message digest.
– Here each of A,B,C,D, is a 32 bit register.
- These registers are initialized to the following values in hexadecimal:

word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

Four auxiliary functions

In addition MD5 uses four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word. They apply the logical operators and, or, not and xor to the input bits.

Round 1 = (b and c) or ((not(b) and d))
Round 2 = (b and d) or (c and not(d))
Round 3 = B xor c xor d
Round 4 = C xor (b or not(d))

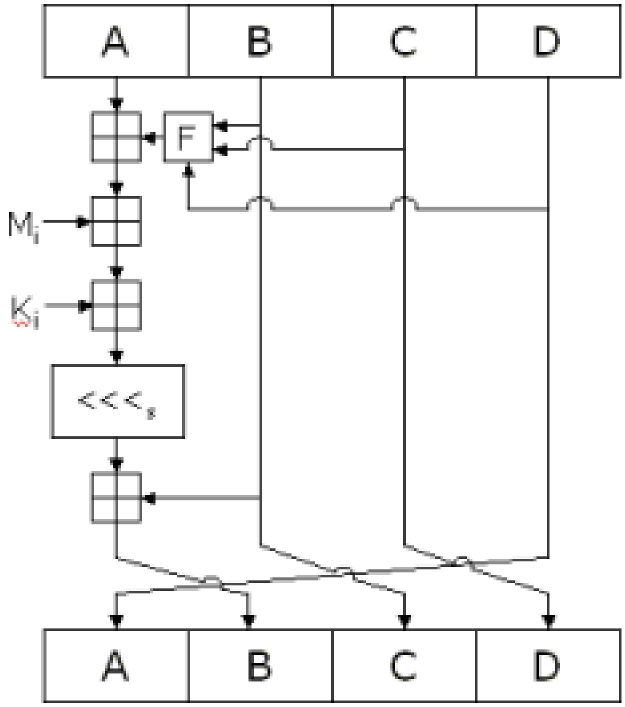
The Constant t[i] or k[i]

MD5 further uses a table K that has 64 elements. Element number i is indicated as K_i. The table is computed beforehand to speed up the computations. The elements are computed using the mathematical sin function:

$K_i = abs(sin(i + 1)) * 2^{32}$

• Step 5 – Process message in 16-word blocks.

1. – Process message in 16-word (512-bit) blocks:
   - – Using 4 rounds of 16 bit operations on message block & buffer
   - – Add output to buffer input to form new buffer value
2. Output hash value is the final buffer value
3. The contents of the four buffers (A, B, C and D) are now mixed with the words of the input, using the four auxiliary functions (F). There are four *rounds*, each involves 16 basic *operations*. One operation is illustrated in the figure below.



The figure shows how the auxiliary function F is applied to the four buffers (A, B, C and D), using message word M_i and constant K_i. The item "<<<s" denotes a binary left shift by *s* bits.

Round 1.
[abcd k s i] denote the operation a = b + ((a + F (b, c, d) + X [k] + T [i]) <<< s).

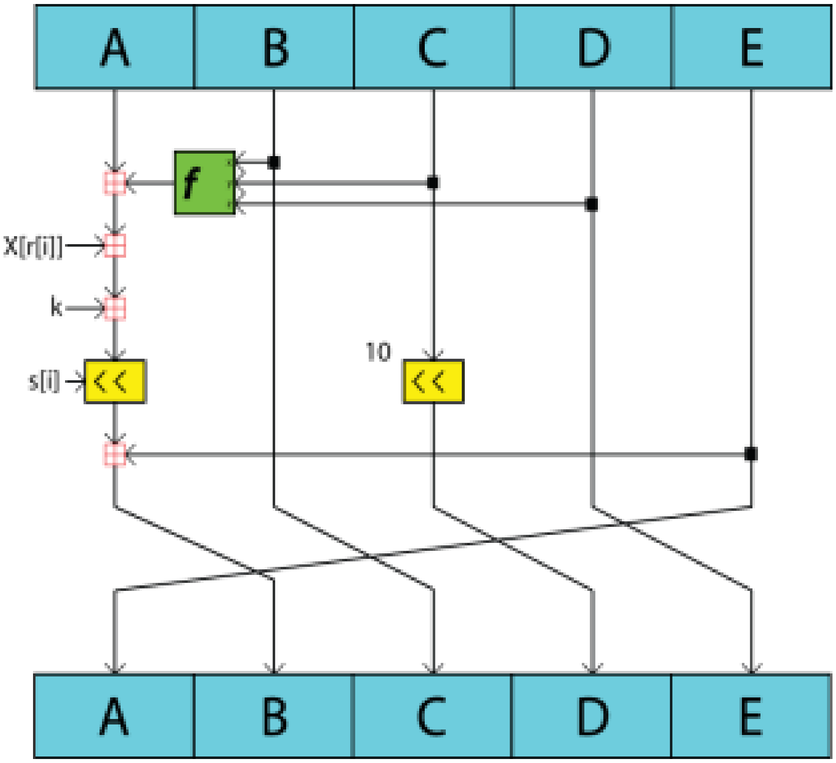
Do the following 16 operations.
[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4] [ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8] [ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12] [ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

Output:

– The message digest produced as output is A, B, C, D.
– That is, output begins with the low-order byte of A, and end with the high-order byte of D.

SHA-1

Processing is similar to SHA-1 with small variations. In SHA-1, chaining variables are 5 and Boolean operations are different.



Analysis

Differences between MD5 and SHA Algorithms

Keys For Comparison	**MD5**	**SHA**
Security	Less Secure than SHA	High Secure than MD5
Message Digest Length	128 Bits	160 Bits
Attacksrequired to find out original Message	2128 bit operations required to break	2160 bit operations required to break

Attacks to try and find two messages producing the same MD	264 bit operations required to break	280 bit operations required to break
Speed	Faster, only 64 iterations	Slower than MD5, Required 80 Iterations
Successful attacks so far	Attacks reported to some extents	No such attach report yet

MD5 Execution

Test Strings	**MD5**	**SHA-1**
1234567890	f807f1fcf80d030febe008fa1708e1ef 31	
abcdefghijklm nopqrstuvwxyz	f3fcf3f711e2f4001dfb191cfa17f10b 15	
message digest	f91b191d1ce7e3ed121a0f01eaf111f0 15	

Timing comparison between MD5 and SHA-1

File Size	**MD5**	**SHA-1**
1 KB		
5 KB		
10 KB		

Practical and Real Time Applications

- In Windows OS, PowerShell function "Get-FileHash"
- Android ROMs

- File servers - file servers often provide a pre-computed MD5 (known as md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it.
- Most unix-based operating systems include MD5 sum utilities in their distribution packages

Conclusion:

The program was tested for different sets of inputs.
Program is working SATISFACTORY NOT
SATISFACTORY(Tick appropriate outcome)

Post Lab Assignment:

1. Why is SHA-1 more secure than MD5?

SHA-1 is more secure than MD5 because it produces a longer hash value (160 bits for SHA-1 vs. 128 bits for MD5). The longer hash size makes it harder to crack it through brute force.

2. Which of the following is not included in hash function?

    a. Authentication.
    b. Message integrity.
    c. Fingerprinting.
    d. Inefficiency.

d. Inefficiency.

3. Which of the following is used to detect transmission errors, and not to detect intentional tampering with data?

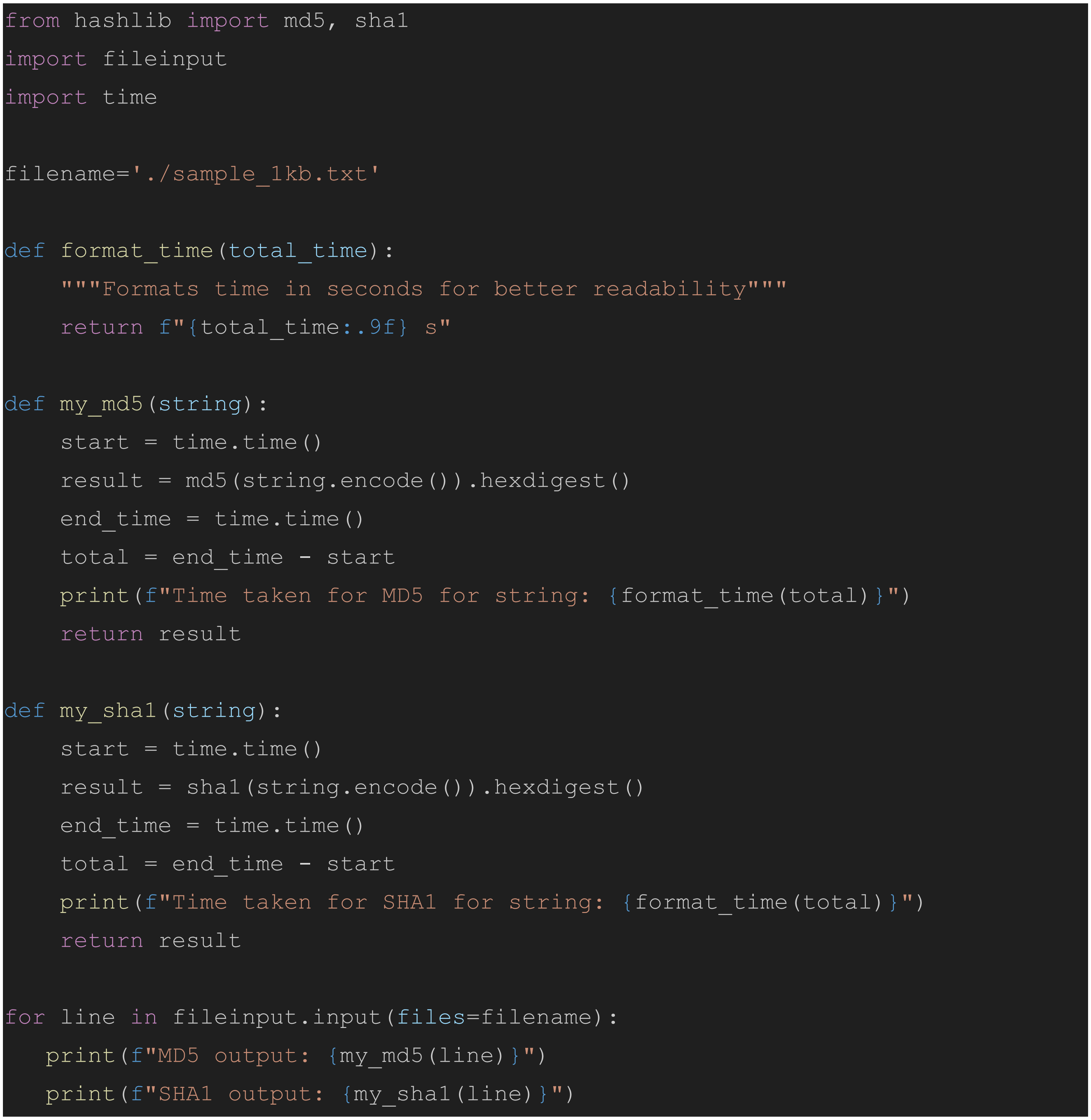
    a. CRC.
    b. Similar checksum.
    c. WEP.
    d. Hash function.

a. CRC.

4. Which of the following is not provide by hash function?

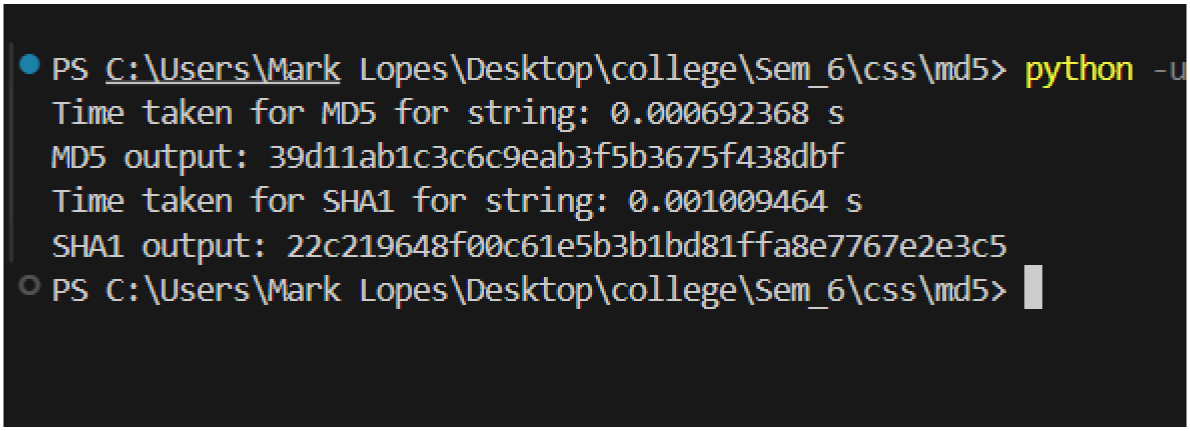
    a. Efficiency.
    b. Two-way.
    c. Compression.
    d. Weak collision resistance.

b. Two-way



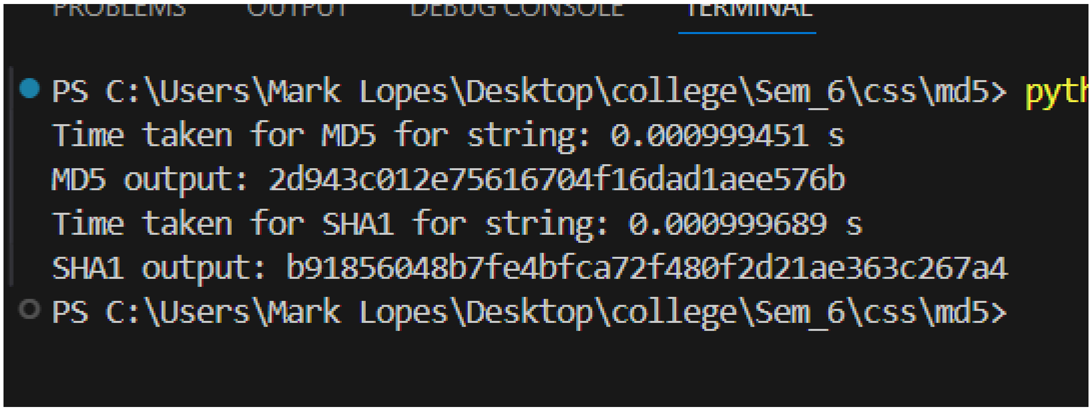
```python
from hashlib import md5, sha1
import fileinput
import time

filename='./sample_1kb.txt'

def format_time(total_time):
    """Formats time in seconds for better readability"""
    return f"{total_time:.9f} s"

def my_md5(string):
    start = time.time()
    result = md5(string.encode()).hexdigest()
    end_time = time.time()
    total = end_time - start
    print(f"Time taken for MD5 for string: {format_time(total)}")
    return result

def my_sha1(string):
    start = time.time()
    result = sha1(string.encode()).hexdigest()
    end_time = time.time()
    total = end_time - start
    print(f"Time taken for SHA1 for string: {format_time(total)}")
    return result

for line in fileinput.input(files=filename):
   print(f"MD5 output: {my_md5(line)}")
   print(f"SHA1 output: {my_sha1(line)}")
```

For 1kb



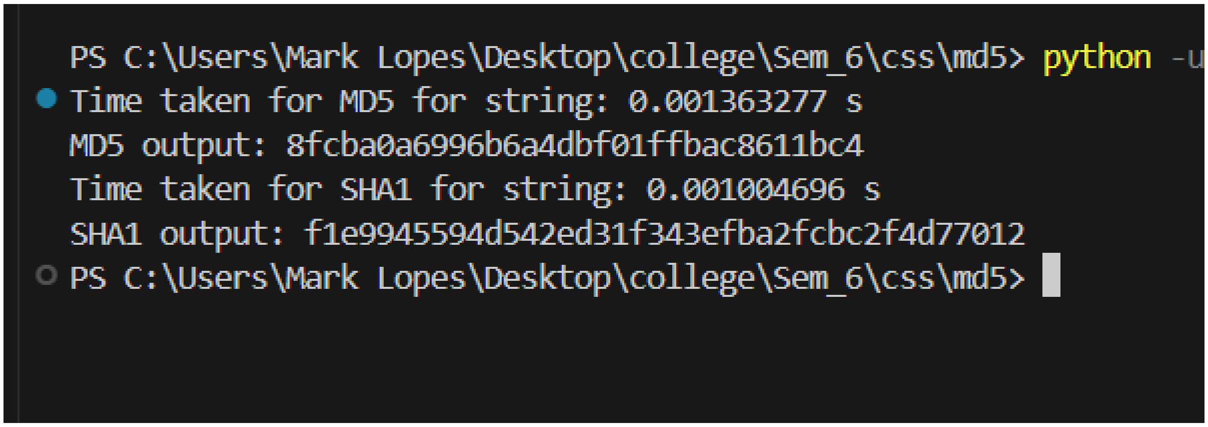
```
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5> python -u
Time taken for MD5 for string: 0.000692368 s
MD5 output: 39d11ab1c3c6c9eab3f5b3675f438dbf
Time taken for SHA1 for string: 0.001009464 s
SHA1 output: 22c219648f00c61e5b3b1bd81ffa8e7767e2e3c5
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5>
```

For 5kb



```
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5> pyth
Time taken for MD5 for string: 0.000999451 s
MD5 output: 2d943c012e75616704f16dad1aee576b
Time taken for SHA1 for string: 0.000999689 s
SHA1 output: b91856048b7fe4bfca72f480f2d21ae363c267a4
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5>
```

For 10kb



```
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5> python -u
Time taken for MD5 for string: 0.001363277 s
MD5 output: 8fcba0a6996b6a4dbf01ffbac8611bc4
Time taken for SHA1 for string: 0.001004696 s
SHA1 output: f1e9945594d542ed31f343efba2fcbc2f4d77012
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\css\md5>
```