

**Name: Deep Kothari**

**Roll No: 23**

**Class: TE COMP B**

### **Experiment No 6**

**Aim:** To analyze the performance and implement for varying message sizes, test integrity of message using MD-5, SHA-1 using crypt APIs

#### **Theory:**

##### **MD5**

The MD5 hash function produces a 128-bit hash value. It was designed for use in cryptography, but vulnerabilities were discovered over the course of time, so it is no longer recommended for that purpose. However, it is still used for database partitioning and computing checksums to validate files transfers.

#### **Algorithm**

##### *Step1: Append Padding Bits*

- Padding means adding extra bits to the original message. So in MD5 original message is padded such that its length in bits is congruent to 448 modulo 512. Padding is done such that the total bits are 64 less, being a multiple of 512 bits length.
- Padding is done even if the length of the original message is already congruent to 448 modulo 512. In padding bits, the only first bit is 1, and the rest of the bits are 0.

##### *Step 2: Append Length*

After padding, 64 bits are inserted at the end, which is used to record the original input length. Modulo  $2^{64}$ . At this point, the resulting message has a length multiple of 512 bits.

*Step 3: Initialize MD buffer.*

A four-word buffer (A, B, C, D) is used to compute the values for the message digest. Here

A, B, C, D are 32-bit registers and are initialized in the following way

Word A	01	23	45	67
Word B	89	Ab	Cd	Ef
Word C	Fe	Dc	Ba	98
Word D	76	54	32	10

*Step 4: Processing message in 16-word block*

MD5 uses the auxiliary functions, which take the input as three 32-bit numbers and produce 32-bit output. These functions use logical operators like OR, XOR, NOR.

F(X, Y, Z)	$XY \vee \text{not}(X)Z$
G(X, Y, Z)	$XZ \vee Y \text{not}(Z)$
H(X, Y, Z)	$X \text{ xor } Y \text{ xor } Z$
I(X, Y, Z)	$Y \text{ xor } (X \vee \text{not}(Z))$

## SHA-1

SHA stands for Secure Hash Algorithm. The first version of the algorithm was SHA-1, and was later followed by SHA-2 (see below).

Whereas MD5 produces a 128-bit hash, SHA1 generates 160-bit hash (20 bytes). In hexadecimal format, it is an integer 40 digits long. Like MD5, it was designed for

cryptology applications, but was soon found to have vulnerabilities also. As of today, it is no longer considered to be any less resistant to attack than MD5.

**Code:**

**MD-5:**

```
# Python code to illustrate the working of MD 5 algorithm to generate
hexadecimal equivalent

import hashlib
#input string
string = input("Enter a string: ")

# encoding the string using encode()
enc = string.encode()

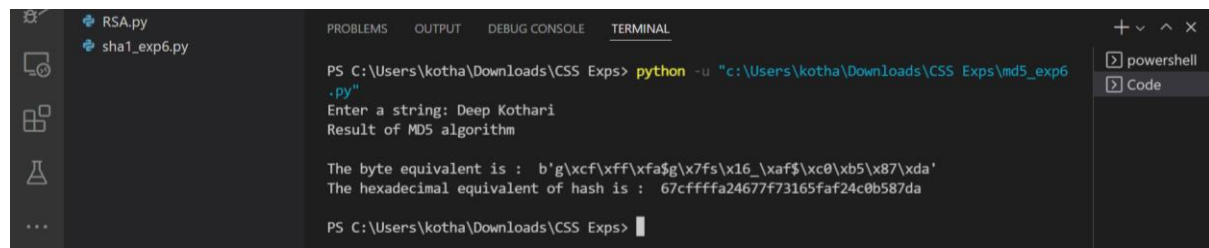
# passing the encoded string to MD5
h = hashlib.md5(enc)

print("Result of MD5 algorithm \n")

# printing the byte equivalent value
print("The byte equivalent is : ", h.digest())

# printing the equivalent hexadecimal value
print("The hexadecimal equivalent of hash is : ", h.hexdigest(), "\n")
```

**Output:**



```
PS C:\Users\kotha\Downloads\CSS Exps> python -u "c:\Users\kotha\Downloads\CSS Exps\md5_exp6.py"
Enter a string: Deep Kothari
Result of MD5 algorithm

The byte equivalent is : b'g\xcf\xff\xfa$g\x7fs\x16_\xaf$\xc0\xb5\x87\xda'
The hexadecimal equivalent of hash is : 67cffffa24677f73165faf24c0b587da

PS C:\Users\kotha\Downloads\CSS Exps>
```

**Code:**

**SHA1**

```
# import the library module
import hashlib
```

```
# initialize a string
str = input("Enter a string: ")

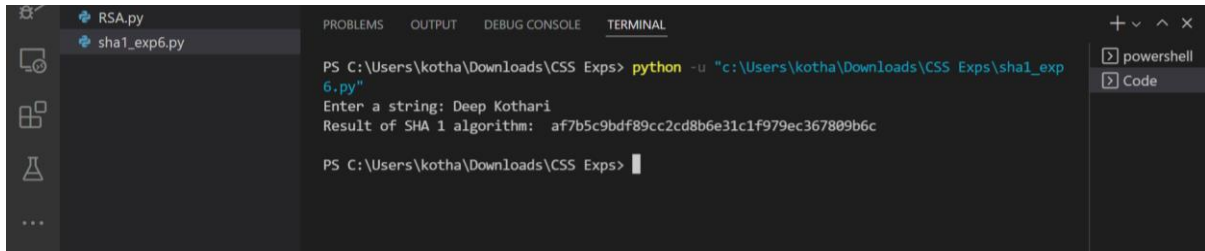
# encode the string
enc = str.encode()

# create a sha1 hash object initialized with the encoded string
hash_obj = hashlib.sha1(enc)

# convert the hash object to a hexadecimal value
hexa_value = hash_obj.hexdigest()

# print
print("Result of SHA 1 algorithm: ", hexa_value, "\n")
```

### Output:



```
PS C:\Users\kotha\Downloads\CSS Exps> python -u "c:\Users\kotha\Downloads\CSS Exps\sha1_exp6.py"
Enter a string: Deep Kothari
Result of SHA 1 algorithm: af7b5c9bdf89cc2cd8b6e31c1f979ec367809b6c
PS C:\Users\kotha\Downloads\CSS Exps>
```

**Conclusion:** We were able to implement MD5 and SHA1 using python programming. It generates digest successfully.