

Terna Engineering College

Computer Engineering Department

Class: TE

Sem.: VI

Course: System Security Lab

PART A

(PART A : TO BE REFERRED BY STUDENTS)

Experiment No.04

A.1 Aim: For varying message sizes, test integrity of message using MD-5, SHA-1, and analyze the performance of the two protocols. Use crypt APIs

A.2 Prerequisite:

1. Basic Knowledge of MD5 and SHA 1

A.3 Outcome:

After successful completion of this experiment students will be able to

To analyze and evaluate performance of hashing algorithms.

A.4 Theory:

MD5 (Message Digest algorithm 5) is a widely used cryptographic hash function with a 128 bit hash value. MD5 hash is typically expressed as a 32 digit hexadecimal number. MD5 processes a variable length message into a fixed length output of 128 bits. The input message is broken up into chunks of 512 bit blocks (sixteen 32bit little endian integers) ; The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64bit integer representing the length of the original message, inbits.

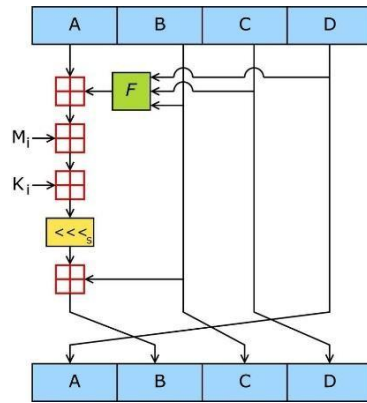


Figure 1: One MD5 operation.

MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. F is a nonlinear function; one function is used in each round. M_i denotes a 32bit block of the message input, and K_i denotes a 32bit constant, different for each operation.

The main MD5 algorithm operates on a 128bit state, divided into four 32bit words, denoted A , B , C and D . These are initialized to certain fixed constants. The main algorithm then operates on each 512bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a nonlinear function F , modular addition, and left rotation.

Figure 1 illustrates one operation within a round. There are four possible functions F ; a different one is used in each round:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

\oplus , \wedge , \vee , \neg denote the XOR, AND, OR and NOT operations respectively.

Algorithm:

1. Append PaddingBits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

2. Append Length

A 64 bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^{64} , then only the low order 64 bits of b are used. (These bits are appended as two 32 bit words and appended low-order word first in accordance with the previous conventions.) At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32 bit) words. Let $M[0 \dots N]$ denote the words of the resulting message, where N is a multiple of 16.

3. Initialize MDBuffer

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, low order bytes first):

4. Process Message in 16 Word Blocks

We first define four auxiliary functions that each take as input three 32 bit words and produce as output one 32 bit word.

Note: Do not write code for MD5 or SHA analyze the performance using crypt APIs

PART B

(PART B : TO BE COMPLETED BY STUDENTS)


(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

| | |
|---------------------|---------------------|
| Roll No.B30 | Name: Pranjal Bhatt |
| Class :COMPS TE B | Batch :B2 |
| Date of Experiment: | Date of Submission |
| Grade : | |

B. Output / Observations

1

Encrypt

 MD5 Encrypt/Decrypt

[Share](#) [Add to Favs](#) [Report Bug](#)

Encrypter

Decrypter

Text

P@30

>>

MD5 Hash

0707cd6d6c33f4124bce0876b90f7c37

Encrypt >

Reset

Copy

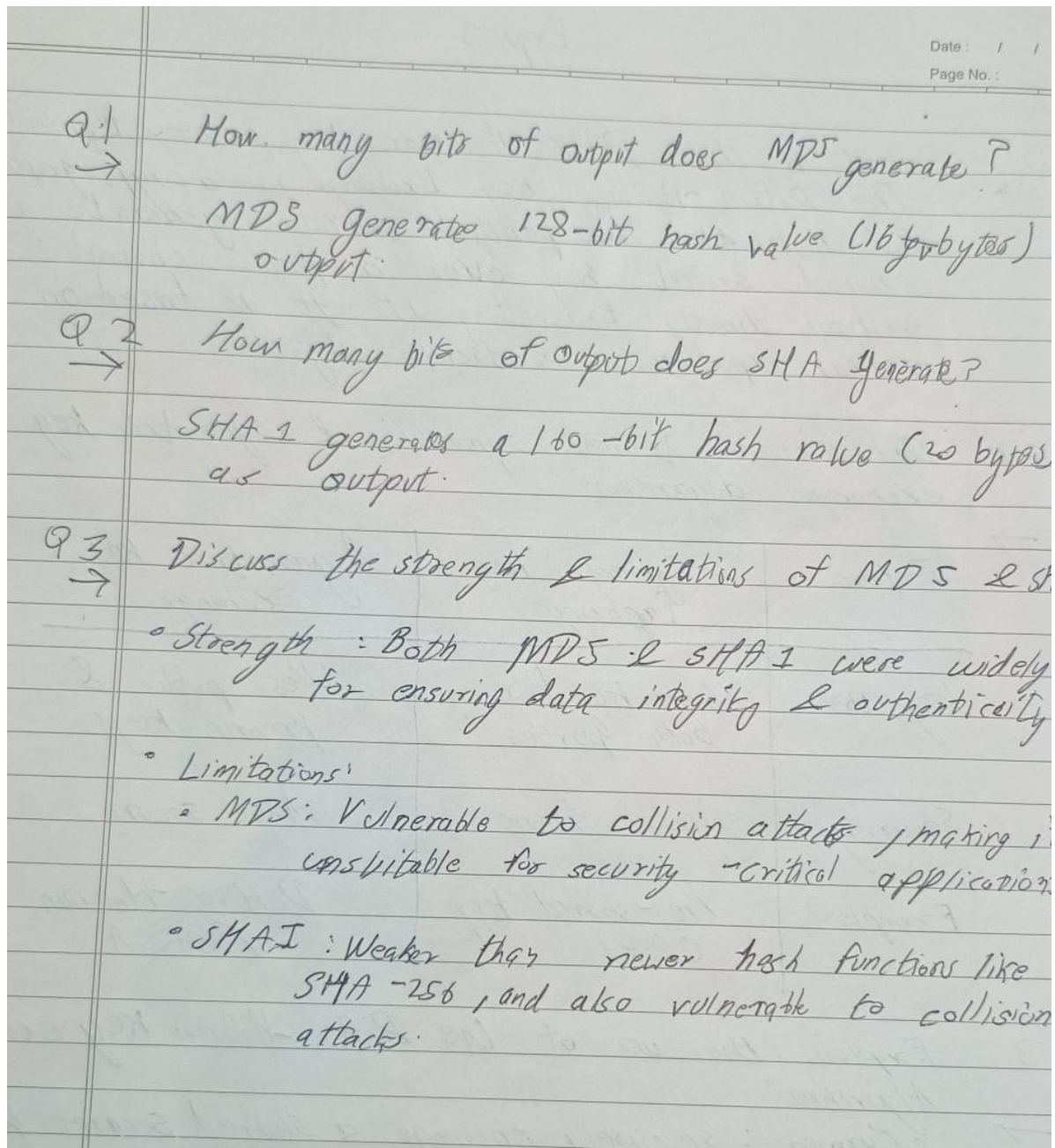
Decrypt

The screenshot shows a web application titled "MD5 Encrypt/Decrypt" with a lock icon. At the top right are buttons for "Share", "Add to Favs", and "Report Bu". Below the title is a large light blue rectangular area. In the center, there are two tabs: "Encrypter" and "Decrypter", with "Decrypter" being the active tab. The interface is divided into two main text input areas. The left area, labeled "MD5 Hash", contains the text "0707cd6d6c33f4124bce0876b90f7c37". The right area, labeled "Text", contains the text "P@30". Between these two areas is a double arrow icon pointing right. At the bottom of the interface, there is a green status bar showing "Elapsed Time: 4.56s" and "Trial Count: 5.3M". Below the status bar are four buttons: "Decryption Settings" with a dropdown arrow, "Decrypt" with a right arrow, "Reset", and "Copy".

B. Question of Curiosity: *(At least 3 questions should be handwritten)*

2

1. How many bits of output MD5 generate?



2. How SHA512 differs from other variants of SHA?

SHA-512 is part of the SHA-2 (Secure Hash Algorithm 2) family and differs from other SHA variants primarily in terms of the output size and the number of rounds used in the hashing process.

Here's how SHA-512 differs from other variants:

1. Output Length:

- **SHA-512** produces a 512-bit hash value (64 bytes).
- Other variants of SHA, like **SHA-256**, produce a 256-bit hash value (32 bytes), and **SHA-1** produces a 160-bit hash (20 bytes).

2. Number of Rounds:

- **SHA-512** uses 80 rounds of processing.
- **SHA-256** uses 64 rounds.
- The number of rounds affects the security and computational complexity, with more rounds generally increasing security.

3. Block Size:

- **SHA-512** processes data in 1024-bit (128-byte) blocks.
- **SHA-256** uses 512-bit (64-byte) blocks.
- Larger block sizes (like in SHA-512) can contribute to better security, as it requires more data to be processed in each round.

4. Word Size:

- **SHA-512** uses 64-bit words during its computation, making it more suitable for 64-bit architectures.
- **SHA-256** uses 32-bit words, which is more efficient on 32-bit systems.

5. Performance:

- On 64-bit systems, **SHA-512** is generally faster than **SHA-256** due to the larger word size and more efficient processing.
- On 32-bit systems, **SHA-256** might perform better, as it's optimized for 32-bit word sizes.

In summary, **SHA-512** differs from other SHA variants mainly in its hash length, the number of rounds, and the word/block size, with these factors impacting both performance and security.

3. How does the collision resistance property differ between MD5 and SHA1?

Collision resistance refers to the property of a cryptographic hash function where it is computationally infeasible to find two distinct inputs that produce the same hash output. In terms of **MD5** and **SHA-1**, their collision resistance differs significantly:

1. MD5 (Message Digest Algorithm 5):

- **Collision resistance** in MD5 is considered **weak**. Over time, researchers have discovered several methods to find collisions in MD5 much faster than brute-forcing all possibilities.
- **Practical vulnerability**: Since 2004, practical collision attacks against MD5 have been demonstrated. These attacks allow an attacker to create two different inputs that hash to the same MD5 value.
- **Current Status**: MD5 is no longer considered secure for cryptographic purposes, especially for applications like digital signatures or certificates, as its collision resistance is broken.

2. SHA-1 (Secure Hash Algorithm 1):

- **Collision resistance** in SHA-1 is stronger than MD5, but it is still **vulnerable**. In 2005, theoretical weaknesses were identified, and since then, several research papers have demonstrated ways to find collisions more efficiently.
- **Practical vulnerability**: In 2017, Google and CWI Amsterdam successfully demonstrated a practical collision attack on SHA-1, called the **SHAttered** attack.
- **Current Status**: SHA-1 is also considered insecure for most cryptographic applications, and it has been phased out for use in digital certificates, code signing, and other security protocols.

Comparison of Collision Resistance:

- **MD5**: Strongly vulnerable to collisions, with practical attacks demonstrated years ago.
- **SHA-1**: More resistant than MD5 but still vulnerable, with practical attacks demonstrated more recently (2017).

4. What are the practical applications of MD5 and SHA1 in modern cryptographic systems?

MD5 and **SHA-1** have limited use today due to their security vulnerabilities, but they are still found in legacy systems.

MD5:

- **Checksums**: Used for simple file integrity checks, such as in file downloads.
- **Password Hashing**: Found in older systems, but not recommended due to security risks.

SHA-1:

- **Digital Signatures:** Previously used in SSL certificates and digital signatures, though now being replaced by stronger algorithms.
- **Git:** Used for version control, but there are efforts to transition to more secure hash functions.

Both are being phased out in favor of **SHA-256** and other secure algorithms.

B. Conclusion:

4

In this experiment, we tested the integrity of messages with varying sizes using **MD5** and **SHA-1** hash functions. We observed that both algorithms are capable of providing hash values that ensure data integrity, but their performance and security differ significantly.

- **MD5** is faster for small messages but is considered **weak** in terms of collision resistance and not recommended for security-critical applications.
- **SHA-1** offers better security than MD5 but still has vulnerabilities, and its performance is slower compared to MD5.

Both algorithms are being phased out in favor of more secure hashing functions like **SHA-256**. Using modern cryptographic APIs allows easy implementation, but for future-proof systems, it is advised to use stronger hash algorithms, especially for large messages or security-sensitive applications.