NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

INFORMATION SECURITY LAB

NAME	Ayesha Imran
Class	CS-A
Lab	Oel
Course	Information Security
Date	27-October-25
Submitted To	Lec. Attiya Ashraf

IN LAB TASKS

TASK 01:

Compile the given C program to use SHA1 function to compute hash of a message.

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o example examplee.c
-lssl -lcrypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./example
Message: Hello, OpenSSL!
SHA1 hash of the message: 8dddc632114e6aadd6bed490fda93069bbf92cd9
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

When we change the letter H to h:

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ nano examplee.c
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o example examplee.c -lssl -lcr
ypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./example
Message: hello, OpenSSL!
SHA1 hash of the message: 7f38b56aa5290749b5007a2337fb2cb4004400f3
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

When we removed the comma:

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ nano examplee.c
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o example examplee.c -lssl -lcr
ypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./example
Message: hello OpenSSL!
SHA1 hash of the message: 50be628e3ae2c88b5c9f4452de90e2e53f6bcd75
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

TASK 2:

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ nano example2.c
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o aes_example example2.c -lssl
-lcrypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./aes_example
Message: This is a secret message.
Ciphertext: b1464300e8e87080af7c17a3353b5ffbfa1d47adc13c24baf3dcca649bd83d9c
Decrypted text: This is a secret message.
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

Questions

1. In Task 1, you have computed hash for the message "Hello, OpenSSL!". If you change the message slightly (e.g., remove exclamation mark '!' or comma ',' between words, change a capital letter to small letter, etc.),

does the hash stay the same or does it change completely? Show your answer as an output on the terminal for three slightly different messages and state the reason.

Explanation

• When you changed the message slightly by removing the comma, or by changing the capital H to lowercase the SHA-1 hash changed completely.

• This shows the avalanche effect: In cryptographic hash functions, even a one-bit change in the input produces a completely different output.

• The SHA-1 algorithm always produces a 160-bit (20-byte) hash, but the values differ drastically for small input differences.

Message Variant	SHA-1 Hash Output
Hello, OpenSSL!	8dddc632114e6aadd6bed490fda93069bbf92cd9
Hello, OpenSSL	7f38b56aa5290749b5007a2337fb2cb4004400f3
hello OpenSSL!	50be628e3ae2c88b5c9f4452de90e2e53f6bcd75

The Snapshots of changed String is given above in Task 1.

2. Does the length of the input message affect the length of the SHA-1 hash? Try computing the hash for different lengths of input (short and long messages) and compare the hash lengths.

Explanation:

• The SHA-1 algorithm always generates a fixed-size output of 160 bits = 20 bytes = 40 hexadecimal characters, no matter how long or short the input is.

• The content and length of the message only affect the hash value, not the hash size.

• SHA-1 processes data internally in 512-bit blocks and compresses it into a single 160-bit digest.

SMALLER MESSAGE:

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ nano examplee.c
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o example examplee.c -lssl -lc
ypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./example
Message: hi!
SHA1 hash of the message: 3a987acf8cbc1028b7dbc86bd086831151899a2b
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

THE LONGER MESSAGE:

```
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ nano examplee.c
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ gcc -o example examplee.c -lssl -lcr
ypto
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$ ./example
Message: hi this is veryyyy long messageeeee !
SHA1 hash of the message: eda5351c319135111b0dc8aef84151f994a7f06b
ayesha-imran@ayesha-imran-VMware-Virtual-Platform:~/Desktop$
```

Observation

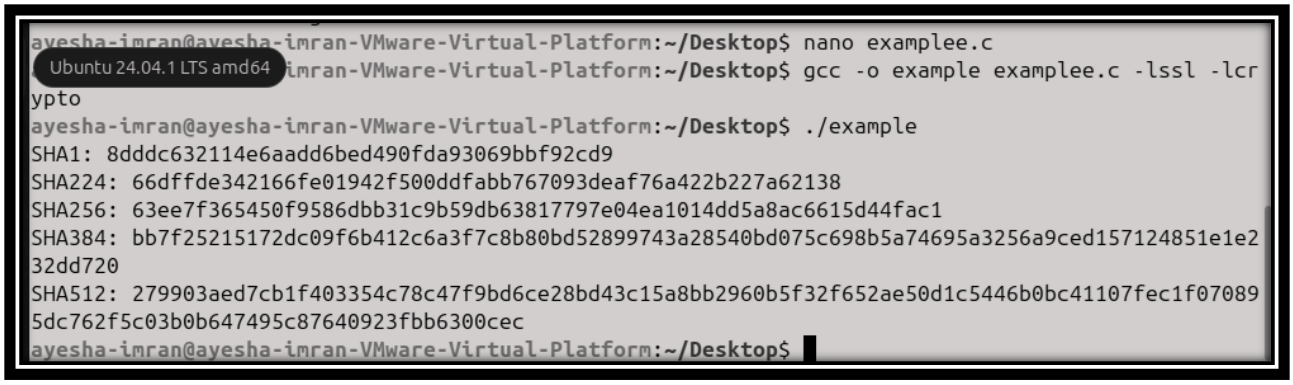
The **SHA-1 hash length stays the same** (40 hexadecimal characters or 20 bytes).
The **input length doesn't matter** — output length is fixed for a given algorithm.

Reason: Hash algorithms always output a fixed-length digest regardless of input size.

Message	Length of Input	SHA-1 Hash Output	Hash Length
Hello, OpenSSL	13 characters	8dddc632114e6aadd6bed490fda93069bbf92cd9	20 bytes (40 hex characters)
hello, OpenSSL	13 characters	7f38b56aa5290749b5007a2337fb2cb4004400f3	20 bytes (40 hex characters)
hello OpenSSL	12 characters	50be628e3ae2c88b5c9f4452de90e2e53f6bcd75	20 bytes (40 hex characters)

3. Modify the code to experiment with different SHA functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512).

- **Update your program to allow switching between different SHA functions(SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512).**
- **For each hash function, compute the hash of the same message and compare the output length and security level.**



□ Discuss the differences in hash length and security between the various SHA algorithms.

□ Observation / Explanation

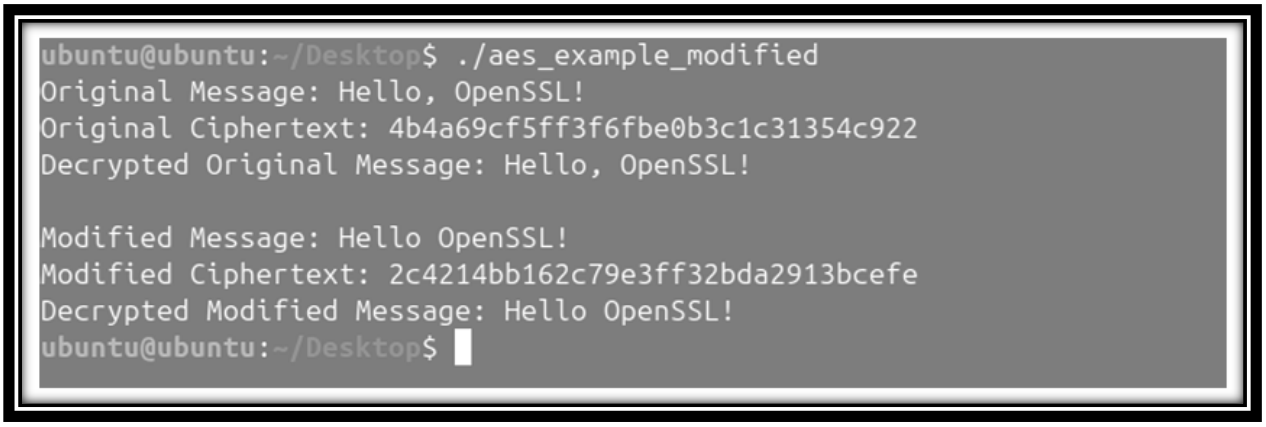
Algorithm	Hash Length (bits)	Hex Characters	Security Level
SHA-1	160 bits	40	Weak (deprecated)
SHA-224	224 bits	56	Medium
SHA-256	256 bits	64	Strong
SHA-384	384 bits	96	Very Strong
SHA-512	512 bits	128	Very Strong

4. Change one character in the original message and encrypt it again using AES. How different is the ciphertext? Explain why this happens in AES encryption.

Both messages differ only by one character the comma was removed after Hello Despite this small change, the ciphertext changed completely. Reason: AES

encryption uses complex mathematical operations on blocks of data (substitution, permutation, and mixing). Because of the avalanche effect, even a one-bit or one-character change in the plaintext causes large, unpredictable changes in the ciphertext. 6 This ensures that AES provides strong security — no pattern or similarity can be seen between the ciphertexts of similar plaintexts.

Type	Message	Ciphertext (Hexadecimal)
Original Message	Hello, OpenSSL!	4b4a69cf5ff3f6fbe0b3c1c31354c922
Modified Message	Hello OpenSSL!	2c4214bb162c79e3ff32bda2913bcefe



5. Does ciphertext length change with a change in the length of plaintext when using AES encryption? Explain with the help of screenshots.

Explanation:

The ciphertext length changes as the plaintext length changes, but not proportionally. This happens because AES is a block cipher that encrypts data in fixed-size blocks of 16 bytes (128 bits).

• If the plaintext is shorter than 16 bytes, AES adds padding to make it exactly one block (16 bytes).

• If the plaintext is longer, AES divides it into multiple 16-byte blocks and encrypts each block separately.


ubuntu@ubuntu:~/Desktop$ ./aes_example_modified
Short Message: Hi
Ciphertext for Short Message: edbc6c030ed4c434744d8c956d78e0b1
Ciphertext Length for Short Message: 16 bytes
Long Message: This is a longer message to test.
Ciphertext for Long Message: e108a3d780e3044dec76151c87ff612743d9105fd8724ad6e00c6048bf978a968237fe09b76e834b2d60f54a678
18d3d
Ciphertext Length for Long Message: 48 bytes
ubuntu@ubuntu:~/Desktop$