Companion Assignment: Encryption and Hashing Results for Ya Zou

Correct answers are hidden.

Score for this attempt: **33** out of 50 *
Submitted Nov 8 at 8:44pm
This attempt took 158 minutes.

In this exercise, you will play with various encryption algorithms and modes. You can use the following OpenSSL enc command to encrypt/decrypt a file. To see the manuals, you can type 'man openssl' and 'man enc'.

```
% openssl enc -base64 [ciphername] -e -in plain.txt -out cipher.txt -K [key] -iv [IV]
```

-base64	Encodes in base64
[ciphername]	Choose cipher type (ex: '-aes-256-cbc')
-in <file>	Input File
-out <file>	Output File
-e	Encrypt Mode
-d	Decrypt Mode
-K & -iv	Key and IV, [value] should be in hexadecimal
-[pP]	Print the Key & IV (then exit if -P)

Question 1

3 / 3 pts

Encrypt the following phrase using -aes-256-cbc with the given Key and IV. Use a TXT file with the phrase as input (no newlines).

"Keep it secret. Keep it safe."

KEY: "bd3dd3bbf6cec43fc354c3bf1a86d0bd84809f8fa93772c8f8b719f0a08c8449"

IV: "7f6989cb19f65b3e6fd13648f4e97fd5"

What is the resulting ciphertext in Base64?

- ○ LvQHLCe0iS40AivhaC70QfgR6rksN0dnTzdSYv9knV0=
- [✓ LvWHLCe0iS40AivhaC70QfgQ6rksN0dnTzdSYv9knV0=]
- ○ 1Yb`q5.N6gdA4E:m&]YRK:gwnoHUUNmY>"G"fyh\Spp!
- ○ A3'VqL>zK}wXNy:)h%SJ7qnYh*r)XnN7Ax`v&gaq:HOn

Question 2

3 / 3 pts

Decrypt the following ciphertext using -aes-256-cbc with the given Key and IV. Use a TXT file with the phrase as input (no newlines).

"HzZMgAKJflTo8jh0PE6GpOR+E1/92TPetEDbrn9RGZQv25W0DgnipbnVMrypCx+wq2PVc0QFzYxxF0AQoGgx0s/JXL/NcJY8Ih0TN9KPu7o="

KEY: "92130686a84beb6235f817078dd2164b834c50780f7a47532547cf16b95e17cd"

IV: "4315f3cfd364472031d5c7178b5e694e"

What is the resulting plaintext in Base 64?

- [✓ 1, 2, 3, 4, 5? That's amazing! I've got the same combination on my luggage!]

Stop saying my passwords. Now Terry's got to update his keychain.

My username is "password" and my password is "password".

The new Wi-Fi password is fourwordsalluppercase.

Question 3

3 / 3 pts

Encrypt the following phrase using -camellia-256-ecb with the given Key. Use a TXT file with the phrase as input (no newlines).

"It's an older code, sir, but it checks out."

KEY: "a5fee3dfcd1b53c769629c9b5a698b57810c14eed29f26dc4ed09be532e9dfe3"

What is the resulting ciphertext in Base64?

[✓ 8LLGkf9DP2SsISiqik3w2JO0oUMH4IjttdpmDdY53IQ7J1ya8fuG01MG4hejUqOS]

M9GiSpsCVjWL6dUT0y2QEXLNHOdVOkq8qW53FhuJQQ6f3UdTPn9iKb9rQVEKOzz8

1rNZEyy3ofAoNna1Cj1erO2skTh9Zk0ZdGhtKlwEpcOvqTTr0LUv8SJgoAoUYM8H

8LLGkf9DP2SsISiqik3w2JO0oUMH4IjttdpmDdY53IQ7J1na8fuG01MG4hejUqOS

Question 4

3 / 3 pts

Decrypt the following ciphertext using camellia-256-ecb with the given Key. Use a TXT file with the phrase as input (no newlines).

"kHt2gCI9B7+dqPECOQyRyZdJi9GucRrpHHF/q9BRqMkJSB0wMQaGIc1m4OKofrob"

KEY:
"9b13f2ca7fdae3e7319ec4c09cd4a8db9eb8d55b484b9ddf5e3b1da9eedf8dce"

What is the resulting plaintext in Base 64?

- ○ The five boxing wizards jump quickly.
- ○ The quick brown fox jumps over a lazy dog.
- [✓ Sphinx of black quartz, judge my vow.]
- ○ How vexingly quick daft zebras jump!

What are the standard block encryption algorithm(s) for the following parts of the world:

Question 5

2 / 2 pts

USA

- [✓ Triple DES]
- ☐ MISTY1
- ☐ Camellia
- [✓ AES]

Question 6

2 / 2 pts

Japan

- ✓ Camellia
- MISTY1
- ✓ AES
- SM4

Question 7

2 / 2 pts

Europe

- ✓ Camellia
- ✓ SHACAL-2
- ✓ AES
- ✓ MISTY1

Question 8

2 / 2 pts

China

- AES

[✓ SM4]

Camellia

MISTY1

IVs, or initialization vectors, act as a random seed to the block cipher modes of encryption, like CBC. In this task you will understand the role of IVs.

- Step 1: Create any file with one line of text and name it plain.txt, the content of the file doesn’t matter.
- Step 2: Now, encrypt the file using aes-128-cbc scheme and write the output to the file ‘encrypt1.txt’.
    - % openssl enc -base64 ciphername -e -in plain.txt -out cipher.txt -K [Key] -iv [IV]
    - You are free to choose the Key and IV in this step, but note them down. Remember, these values need to be in hexadecimal.
- Step 3: Now using the same Key and IV in the step 2, encrypt the original plaintext file again and write the output to ‘encrypt2.txt’.
- Step 4: Now using the same Key in Step 2 and a different IV which differs only by one bit, encrypt the original plaintext file again and write your output to ‘encrypt3.txt’.

Question 9

2 / 2 pts

Do contents of 'encrypt1.txt' and 'encrypt2.txt' match?

[✓ True]

False

Question 10

Not yet graded / 2 pts

Explain why the contents of 'encrypt1.txt' and 'encrypt2.txt' do or do not match (i.e., explain your answer above)?

Your Answer:

The contents of 'encrypt1.txt' and 'encrypt2.txt' will match because in both steps 1 and 2, we use the same key, initialization vector, and the same original text to do the encryption using the same method (AES-128-CBC). So, 'encrypt1.txt' and 'encrypt2.txt' will end up having the same content.

Question 11

2 / 2 pts

Do the contents of 'encrypt1.txt' and 'encrypt3.txt' match?

True [✓ False]

Question 12

Not yet graded / 2 pts

Explain why the contents of 'encrypt1.txt' and 'encrypt3.txt' do or do not match (i.e., explain your answer above)?

Your Answer:

"encrypt1.txt" and "encrypt3.txt" will not match. In Step 3, it's mentioned that a different initialization vector with just one bit changed is used to encrypt the text files in CBC mode. Even if the same key is used for encryption, the contents won't be the same.

Question 13

Not yet graded / 2 pts

How different, if at all, are the contents of 'encrypt1.txt' and 'encrypt3.txt'. Explain your observation.

Your Answer:

The contents of 'encrypt1.txt' and 'encrypt3.txt' will be entirely different due to the use of different Initialization Vectors (IVs), even though they share the same encryption Key. This is a fundamental property of block cipher modes like CBC.

In this exercise, you will generate message digests using a one-way hash algorithms.

You can use the following openssl dgst command to generate the hash value for a file.

```
% openssl dgst dgsttype filename
```

Please replace the dgsttype with the specific one-way hash algorithm.

To see the manuals, you can type 'man openssl' and 'man dgst'.

Question 14

3 / 3 pts

Hash the following phrase using SHA-256.

"There has never been a sadness that can't be cured by breakfast food."

What is the resulting hash?

○ 29vk6xf7ajoz0f2umkmtdq9el8a0xwtxmhfc1hvzl4obvepnqb9ogj3nz55mhl6p

- ○ 46a0ac881b337554cc06dd4118eee5f5a60c34333284a2ab9ae16bee4247eb42
- ○ 5zl6foduc9raegyf33pvr533yf1u45us5qpzqpghfinu508igpejoz37wdu707ax
- [✓ 46a0ac881b337554cc06dd4118eee5f5a60c34333284a2ab9ae16bea4247eb42]

In this exercise, you will generate a keyed hash (i.e. MAC) for a file.

Please generate a keyed hash using HMAC-SHA256 and HMAC-SHA1 for any file that you choose. Please try several keys with different lengths.

You can use the -hmac option of openssl command line.

The following example generates a keyed hash for a file using the HMAC-SHA256 algorithm. The string following the -hmac option is the key.

```
% openssl dgst –sha256 -hmac "abcdefg" filename
```

Question 15

2 / 2 pts

One has to use a specific fixed-size key with HMAC

- ○ True
- [✓ False]

Question 16

Not yet graded / 3 pts

If a fixed size key must be used, what is this size? If a fixed size doesn't have to be used, why?

Your Answer:

A fixed size doesn't have to be used because HMAC supports variable-length keys. It's designed to work with keys of various sizes and can accommodate keys of different lengths while maintaining security.

To understand the properties of one-way hash functions, do the following exercises using SHA256 :

1. Create a text file with the following text -- ‘The cow jumps over the moon’
2. Generate the hash value H1 for this file.
3. Change the first hex character of the input file to "44". You can achieve this modification using hex editors like ghex or Bless.
4. Generate the hash value H2 for the modified file.

Hash command:

```
% openssl dgst [dgsttype] [filename]
```

Question 17

2 / 2 pts

Using SHA256, what is the value of the H1 hash?

- ebg9eaimgbtr7djc5me7mfw9kmty554b2rwpesm625dhtr6f7du4t7ukepn2vm8v
- pri3h892pmhz0g6t560f6vxrhfe9gv6r90rlzw7groougvn8aao5wypmq72kc0fz

- ○ aa2f7b095a4e2fb3918fa20b2221544d61632aba9fd9e8afa353eaaefde7s6db
- [✓ aa2f7b095a4e2fb3918fa20b2221544d61632aba9fd9e8afa353eaaefde776db]

Question 18

2 / 2 pts

Using SHA256, what is the value of the H2 hash?

- ○ 9787242dd5ea2f1318c57d09b653e02b35f1c719e24a70bdbc942d44988eb02d
- ○ 03542cbed3ffc44e9e4c27f3a40b9f49be1fc17c2b60331d6047976e12ae11ca
- [✓ 9787242dd5ea2f1318c57d09b653e02b35f1c719e24a45bdbc942d44988eb02d]
- ○ 45f3e92e810c5447177062784e9403abf0cd09748941732af30ef992d4a67b29

Question 19

Not yet graded / 3 pts

Try flipping multiple characters in the original file.

What is the relationship between the number of characters you alter in the input and the number of characters changed in the output if any?

Your Answer:

if I alter one or more characters in the input, it is expected that the number of characters changed in the output (the hash value) will be proportionally different. However, the exact relationship between the number of characters altered in the input and the number of characters changed in the output is not straightforward. It depends on the specific changes you make and how those changes affect the internal state of the hash function during the hashing process. In practice, even a single-bit change in the input can result in a hash value that appears to be completely unrelated to the original input. For example, if you have a hash function that takes an input of 128 bits and produces a hash of 256 bits. If you change one bit of the input, the output hash will change at least 50% of the time. This is because the hash function is designed to be highly sensitive to changes in the input, and even a small change can cause a large change in the output.

Question 20

Not yet graded / 5 pts

To see the cost of generating a key-pair, generate several RSA key pairs of varying sizes. A command to do this is below.

```
% openssl genrsa -out private-key.pem [key_size]
```

Key Sizes: 512, 1024, and 2048, 4096

What can you observe about generating pairs with different key sizes.

Your Answer:

- As I increase the key size, the time required to generate the key pair also increases. This is because a larger key requires more computation to generate. For example, a 4096-bit key will take significantly longer to generate than a 512-bit key, also larger keys also require more storage space. For example, a 4096-bit key will require twice as much storage space as a 2048-bit key.
- The security of RSA encryption depends on the key size. Smaller key sizes, such as 512 and 1024 bits, are considered less secure and may be vulnerable to attacks in the modern computing environment. Larger key sizes, such as 2048 and 4096 bits, offer higher security due to the increased complexity of factoring the keys.

- The key size can also affect the performance of encryption and decryption processes. Larger key sizes may result in slower encryption and decryption operations

Quiz Score: **33** out of 50