Lab 3: OpenSSL Lab

Instruction:

In this lab, you will perform symmetric encryption and hashing utilising one of the most widely used crypto tools called Openssl. It is widely used in Internet web servers, serving a majority of all web sites. OpenSSL contains an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements basic cryptographic functions and provides various utility functions. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available.

In this lab, we will use its different utility functions to perform symmetric encryption and hashing. You will be given a set of tasks for using Openssl to perform these functions. For this, you will also need to use a HEX editor like GHex in Ubuntu. Follow the instructions, complete the tasks and prepare a report as per instructed.

Task – 1: AES encryption using different modes

In this task, we 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 -bf-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

```
% openssl enc ciphertype -e  -in plain.txt -out cipher.bin \
             -K  00112233445566778889aabbccddeeff \
             -iv 0102030405060708
```

Replace the ciphertype with a specific cipher type, such as -aes-128-cbc, -aes-128-cfb, -bf-cbc, etc. You can find the meaning of the command-line options and all the supported cipher types by typing "man enc". We include some common options for the openssl enc command in the following:

```
-in <file>     input file
-out <file>    output file
-e             encrypt
-d             decrypt
-K/-iv         key/iv in hex is the next argument
-[pP]          print the iv/key (then exit if -P)
```

In this task, you should encrypt using AES with three different modes for a given text file. At first, create a text file, add several lines of texts and save it to a preferred location. Complete this task by performing the encryption and generating an output file containing the encrypted output. Perform the decryption of the generated encrypted files to test that the encryption works well.

Task – 2 Encryption mode – corrupted cipher text

To understand the properties of various encryption modes, we would like to do the following exercise:

- Create a text file that is at least 64 bytes long. Use the gedit text editor to create and save the file. gedit also allows you to count the byte number in the file.
- Encrypt the file using the AES-128 cipher.
- Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this corruption using a hex editor - C
- Decrypt the corrupted file (encrypted) using the correct key and IV.

Answer the following questions with the commands used to encrypt and decrypt the file:

(1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively? Answer this question before you conduct this task, and then find out whether your answer is correct or wrong after you finish this task.

(2) Explain why.

(3) What are the implications of these differences?

Task – 3: Encryption mode - ECB vs CBC

The file pic original.bmp contains a simple picture. We would like to encrypt this picture, so people without the encryption keys cannot know what is in the picture. Encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes, and then do the following:

- Let us treat the encrypted picture as a picture, and use a picture viewing software to display it. However, for the .bmp file, the first 54 bytes contain the header information about the picture, we have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. We will replace the header of the encrypted picture with that of the original picture. You can use ghex to directly modify binary files.
- Display the encrypted picture using any picture viewing software. Can you derive any useful information about the original picture from the encrypted picture?

Notebook link: Lab 3 - AES.ipynb [Summer-2025]