Lab 1: OpenSSL Symmetric Encryption Algorithms

1 Overview

The learning objective of this lab is for students to get familiar with the concepts in the symmetric encryption algorithm. After finishing the lab, students should be able to gain first-hand experience on symmetric encryption algorithms, operation modes, paddings, and initial vector (IV). Moreover, students will be able to use tools and write commands to encrypt/decrypt messages and files by using symmetric encryption labs.

2 Lab Environment

In this lab, we will use `openssl` commands such as rand, passwd, enc, genrsa, rsautl, dgst, req, verif

The systanx of openssl command is:

```
openssl command [options] [arguments]
```

Installing a hex editor. In this lab, we need to be able to view and modify files of binary format. You can have install a hex editor such as `GHex`. It allows the user to load data from any file, view and edit it in either hex or ascii.

Lab Tasks In the following, a list of tasks that you should do it.

3 Task 1: Encoding with base64

Definition: Base64 is an encoding scheme which uses 65 printable characters (26 lower-case letters, 26 upper-case letters, 10 digits, characters '+' and '/', and special character '='). Base64 allows to exchange data with limited encoding problems.

To encode with base64, the following command is used:

```
 % openssl enc -base64 -in input-file -out output-file
```

To decode with base64, the following command is used:

```
 % openssl enc -base64 -d -in input-file -out output-file
```

To encode a string such as "Hello from the other side" using Base64:

```
 %echo -n "Hello from the other side" | OpenSSL base64
```

1. Encode a text file containing an arbitrary password with base64, and store it. How can an attacker find your password from this file.
2. Is base64 a safe way to protect a password?
3. Decode a file (Base64) and store the output into another text file called "decoded.data"
4. Encoding the string "Hello from the other side" using Base64

4 Task 2: Working with Random Number Generators

To Generate random bytes in OpenSSL, we use the command rand. You can find the meaning of the command-line options and all the supported options by typing "`openssl rand -help`". We include some common options for the `openssl rand` command in the following:

```
rand [options] num
 -out outfile        Output file
 -writerand outfile  Write random data to the specified file
 -base64             Base64 encode output
 -hex                Hex encode output
```

5. Generate 100 random bytes and show them on screen (hexadecimal representation)
6. Generate 200 random bytes and store them in a file named ”rand.txt”
7. Generate 300 random bytes and Encoding them into Base64 and store it in a file called ”rand.base64”

5 Task 3: Encryption using different ciphers and modes

In this task, we will play with various encryption symmetric encryption algorithms and modes. You can use the following `openssl enc` command to encrypt/decrypt a file. To see the manuals, you can type `openssl enc -help`.

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

Please replace the `ciphertype` with a specific cipher type, such as `-aes-128-cbc`, `-aes-128-cfb`, `-bf-cbc`, etc. To use DES, we can use option -des. To use triple-DES, we can use option -des3. In this task, you should try at least 3 different ciphers and three different modes. You can find the meaning of the command-line options and all the supported cipher types by typing "`openssl enc -help`". 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)
```

To encrypt, we can use command enc. To decrypt, we can use command enc with option -d.

5.1 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. Please encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes, and then do the following:

8. 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 a hex editor tool (e.g. `ghex` or `Bless`) to directly modify binary files.

9. Display the encrypted picture using any picture viewing software. Can you derive any useful information about the original picture from the encrypted picture? Please explain your observations.

5.2 Encryption Mode – Corrupted Cipher Text

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

10. Create a text file that is at least 64 bytes long.

11. Encrypt the file using the AES-128 cipher.

12. , Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this corruption using a hex editor.

13. Decrypt the corrupted file (encrypted) using the correct key and IV.

Please answer the following questions:

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

15. Please explain why.

16. What is the implication of these differences?

5.3 Padding

For block ciphers, when the size of the plaintext is not the multiple of the block size, padding may be required. In this task, we will study the padding schemes. Please do the following exercises:

17. The openssl manual says that openssl uses PKCS5 standard for its padding. Please design an experiment to verify this. In particular, use your experiment to figure out the paddings in the AES encryption when the length of the plaintext is 20 octets and 32 octets.

18. Please use ECB, CBC, CFB, and OFB modes to encrypt a file (you can pick any cipher). Please report which modes have paddings and which ones do not. For those that do not need paddings, please explain why.