# Computer and Network Security: Self Study Lab

**Lab #2-1 Symmetric Encryption (I)**

## 1. Introduction

The learning objective of this self-study lab is for students to get familiar with the concepts and practical usage in the symmetric secret-key encryption using *openssl*. After finishing the lab, students should be able to gain a first-hand experience on encryption algorithms, encryption modes, and padding.

OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) network protocols and related cryptography standards. The *openssl* program is a command line tool for using the various cryptography functions of OpenSSL's *crypto* library from the shell. It can be used for

- Encryption and Decryption with various ciphers
- Calculation of Message Digests
- Creation of RSA, DH and DSA key parameters
- Creation of X.509 certificates, CSRs and CRLs
- SSL/TLS Client and Server Tests
- Handling of S/MIME signed or encrypted mail

## 2. Lab 1: Install `openssl` or check version

Most Linux distributions includes `openssl` command. Latest Linux based distribution of `openssl` can be found at *http://www.openssl.org/source/*. The Windows version can be found at *http://www.slproweb.com/products/Win32OpenSSL.html*.

The `openssl` has the following general syntax:

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

**Lab 1.1:** check `openssl` version, if not exists install it. Show your `openssl` version with the following command line.

```
$ openssl version
OpenSSL 1.0.2k-fips  26 Jan 2017
```

**Lab 1.2:** if your `openssl` version is not up-to-dated. What are the vulnerabilities of your `openssl` version.

## 3. Lab 2: Simple encoding with base64

Base64 is an encoding scheme that represents binary data in an ASCII string format by translating it into a radix-64 representation. It uses 65 printable characters (26 lower-case letters, 26 upper-case letters, 10 digits, characters '+' and '/', and special character '='). Base64 encoding schemes are commonly used when there is a need to encode binary data that needs be stored and transferred over media that are designed to deal with textual data.

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
```

**Lab 2.1:** Study the output format from base64 encoding. Explain the group of bits processing and padding scheme as well as the “==” and “=” meaning.

ฟังก์ชั่นนี้จะเปลี่ยน 6 บิทเป็น 1 char
และใช้ = เป็นตัว padding ในกรณีไม่ครบ

**Lab 2.2:** Create an arbitrary text file and encode the file with base64 and show the result.

```
echo "Helloworld" > test.txt
openssl enc -base64 -in test.txt -out test-base64.txt
cat test-base64.txt
SGVsbG93b3JsZAo=
```

**Lab 2.3:** Select a binary file from the system and encode the file with base64 and show the result.

```
openssl enc -base64 -in ../../../bin/make -out make-base64.txt
cat make-base64.txt
```

## 4. Lab 3: Encoding with DES

**Hints:** Studying the DES and AES encryption methods using openssl from textbook and recommended reading as long as the mode of operations to complete the lab exercises.
**Recommended Reading:**
1. https://wiki.openssl.org/index.php/Command_Line_Utilities
2. http://www.tutonics.com/2012/10/easy-file-encryption-using-openssl.html

**Lab 3.1:** Using command “`ls -al >dir1.txt`” to generate a file `dir1.txt`

**Lab 3.2:** Encode the `dir1.txt` with DES ECB to get the encoded file `dir1.ecb`
*Show the command.*

```
openssl enc -des-ecb -in dir1.txt -out dir1.ecb
```

**Lab 3.2:** Encode the `dir1.txt` with DES CBC to get the encoded file `dir1.cbc`
*Show the command.*

```
openssl enc -des-cbc -in dir1.txt -out dir1.cbc
```

**Lab 3.4:** `openssl` has a standard command rand to generate random number that can be used as a random key with a specific length.
*What is the command to generate 56 bits key to the output file* `des_key`?

```
openssl rand 56 > des key
```

*What is the command to generate 256 bits key to the output file* `aes_key`?

```
openssl rand 256 > aes key
```

**Lab 3.4:** Encode the `dir1.txt` with AES 256 ECB to get the encoded file `dir1.ecb`
*Show the command.*

```
openssl enc -aes-256-ecb -in dir1.txt -out dir1.ecb
```

**Lab 3.5:** Encode the `dir1.txt` with AES 256 CBC to get the encoded file `dir1.cbc`
*Show the command.*

```
openssl enc -aes-256-cbc -in dir1.txt -out dir1.cbc
```

## 5. ECB and CBC

The file `penguin-cartoon-clipart-13.bmp` in class repository 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 and CBC modes, and then do the following:

**Lab 4.1:** 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 the `ghex` tool or other hex editor tool to directly modify binary files.

**Lab 4.2**: 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.

Adapted from :
*http://www.cis.syr.edu/~wedu/seed/Labs/Crypto/Crypto_Encryption/Crypto_Encryption.pdf*