Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 6
Year 2, Semester 1

Crypto Lab – Secret-Key Encryption

1. Overview

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

2. Lab Environment

Installing OpenSSL. In this lab, we will use openssl commands and libraries. We have already in- stalled `openssl` binaries in our VM. It should be noted that if you want to use `openssl` libraries in your programs, you need to install several other things for the programming environment, including the header files, libraries, manuals, etc. We have already downloaded the necessary files under the directory `/seed/openssl-1.0.1e`. To configure and install `openssl` libraries, run the following commands.

```
You should read the INSTALL file first:

% ./config
% make
% make test
% sudo make install
```

Installing a hex editor. In this lab, we need to be able to view and modify files of binary format. We have installed in our VM a hex editor called GHex. It allows the user to load data from any file, view and edit it in either hex or ascii. Note: some people told us that another hex editor, called Bless, is better; this tool may not be installed in the VM version that you are using, but you can install it yourself.

Install GHex

```
sudo apt-get install ghex
```

Which will install ghex and any other packages on which it depends.

3. Lab Tasks

3.1 Lab Tasks - Part 01

Task 1: Encryption and Decryption using different ciphers and modes

The algorithm seems to follow the pattern:
(Algorithm name)-(key size)-(encryption mode)
Noted: If the key size is omitted or excluded then it means there is only one key-size for that algorithm.

- **Algorithm name:** Sometimes there is a number included in the algorithm name whose usage is to distinguish the version of the algorithm; for instance, RC2 and RC4.
- **Key size:** key size is in a bit. The longer the key the stronger your encryption is, but the slower operation it takes.
- **Encryption mode:** there are five main encryption mode that widely uses in block cipher mode operation, Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR)

There are several encryption algorithms in OpenSSL as shown in the image below.

Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 6
Year 2, Semester 1

```
$ openssl enc --help
-aes-128-cbc              -aes-128-cfb              -aes-128-cfb1
-aes-128-cfb8             -aes-128-ecb              -aes-128-ofb
-aes-192-cbc              -aes-192-cfb              -aes-192-cfb1
-aes-192-cfb8             -aes-192-ecb              -aes-192-ofb
-aes-256-cbc              -aes-256-cfb              -aes-256-cfb1
-aes-256-cfb8             -aes-256-ecb              -aes-256-ofb
-aes128                   -aes192                   -aes256
-bf                       -bf-cbc                   -bf-cfb
-bf-ecb                   -bf-ofb                   -blowfish
-cast                     -cast-cbc                 -cast5-cbc
-cast5-cfb                -cast5-ecb                -cast5-ofb
-des                      -des-cbc                  -des-cfb
-des-cfb1                 -des-cfb8                 -des-ecb
-des-ede                  -des-ede-cbc              -des-ede-cfb
-des-ede-ofb              -des-ede3                 -des-ede3-cbc
-des-ede3-cfb             -des-ede3-cfb1            -des-ede3-cfb8
-des-ede3-ofb             -des-ofb                  -des3
-desx                     -desx-cbc                 -rc2
-rc2-40-cbc               -rc2-64-cbc               -rc2-cbc
-rc2-cfb                  -rc2-ecb                  -rc2-ofb
-rc4                      -rc4-40
```

In this task, we will get to know 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 <cipher algorithm> -e/-d -in <input_file_name> -out
<output_file_name> -K <key> -iv <iv_value>
```

```
    <cipher algorithm> = one of the cypher algorithms above
           -e/-d       = e for encryption and d for decryption
           -K/-iv      = key/iv in hex is the next argument
```

Please replace the `cipher algorithm` with a specific cipher type, such as `-aes-128-cbc`, `-aes-128-cfb`, `-bf-cbc`, etc. 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 "`man enc`".

Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 6
Year 2, Semester 1

Task 2: Encryption Mode – ECB vs. CBC

Encryption – ECB

The Electronic Codebook (ECB) mode is a confidentiality mode that features the message is divided into blocks, and each block is encrypted separately. The Electronic Codebook (ECB) mode is defined as follows (Evans, Bond, & Bement, 2001):

- **ECB Encryption:** $C_j = CIPH_k\ (P_j)$ for j = 1 … n.

  $\boldsymbol{CIPH_k\ (X)}$ The forward cipher function of the block cipher algorithm under the key K applied to the data block X.
- **ECB Decryption:** $P_j = CIPH_k^{-1}(C_j)$ for j = 1 … n.

  $\boldsymbol{CIPH_k^{-1}(X)}$ The inverse cipher function of the block cipher algorithm under the key K applied K to the data block X.

In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of output blocks is the ciphertext.

In ECB decryption, the inverse cipher function is applied directly and independently to each block of the ciphertext. The resulting sequence of output blocks is the plaintext.

With ECB, if the same b-bit block of plaintext appears more than once in the message, it always produces the same ciphertext. Because of this, for lengthy messages, the ECB mode may not be secure.

Encryption – CBC

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an IV to combine with the first plaintext block. The IV need not be secret, but it must be unpredictable; The CBC mode is defined as follows (Evans, Bond, & Bement, 2001):

Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 6
Year 2, Semester 1

- **CBC Encryption:** $C_1 = CIPH_k(P_1 \oplus IV);$

  $C_j = CIPH_k(P_j \oplus C_{j-1})$ for j = 2 … n.

- **CBC Decryption:** $C_1 = CIPH_k^{-1}(C_1) \oplus IV;$

  $C_j = CIPH_k^{-1}(C_j) \oplus C_{j-1}$ for j = 2 … n.

In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.

The below link you can download the Bmp file "`pic original.bmp`" it 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:

1. Let us treat the encrypted picture as a picture, and use picture viewing software to display it. How- ever, 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.
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.

Bmp File:
http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Crypto/Crypto_Encryption/files/pic_original.bmp

Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 6
Year 2, Semester 1

3.2 Lab Tasks - Part 02

Task 3: Encryption Mode – Corrupted Cipher Text

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

1. Create a text file that is at least 64 bytes long.
2. Encrypt the file using the AES-128 cipher.
3. Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this corruption using a hex editor.
4. Decrypt the corrupted file (encrypted) using the correct key and IV.

Please answer the following questions:

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.

(1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?

(2) Please explain why.

(3) What is the implication of these differences?

Task4: 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:

1. 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.

2. 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

Task 5: Programming using the Crypto Library

So far, we have learned how to use the tools provided by `openssl` to encrypt and decrypt messages. In this task, we will learn how to use `openssl`'s crypto library to encrypt/decrypt messages in programs.

OpenSSL provides an API called EVP, which is a high-level interface to cryptographic functions. Al- though OpenSSL also has direct interfaces for each individual encryption algorithm, the EVP library provides a common interface for various encryption algorithms. To ask EVP to use a specific algorithm, we simply need to pass our choice to the EVP interface.

A sample code is given below
https://www.openssl.org/docs/man1.0.2/man3/EVP_EncryptInit.html

Please get yourself familiar with this program, and then do the following exercise.
You are given plaintext and ciphertext, and you know that `aes-128-cbc` is used to generate the ciphertext from the plaintext, and you also know that the numbers in the IV are all zeros (not the ASCII character '0'). Another clue that you have learned is that the key used to encrypt this plaintext is an English word shorter than 16 characters; the word that can be found from a typical English dictionary. Since the word has less than 16 characters (i.e. 128 bits), space characters (hexadecimal value 0x20) are appended to the end of the word to form a key of 128 bits. Your goal is to write a program to find out this key. You can download an English word list from the Internet (link is given below). The plaintext and ciphertext are in the following:

```
Plaintext (total 21 characters): This is a top secret.
Ciphertext (in hex format): 8d20e5056a8d24d0462ce74e4904c1b5
13e10d1df4a2ef2ad4540fae1ca0aaf9
```

English Word Text :
http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Crypto/Crypto_Encryption/files/words.txt

Note 1: If you choose to store the plaintext message in a file, and feed the file to your program, you need to check whether the file length is 21. Some editors may add a special character to the end of the file. If that happens, you can use a hex editor tool to remove the special character.

Note 2: In this task, you are supposed to write your own program to invoke the crypto library. No credit will be given if you simply use the `openssl` commands to do this task.

Note 3: To compile your code, you may need to include the header files in `openssl`, and link to `openssl` libraries. To do that, you need to tell your compiler where those files are. In your `Makefile`, you may want to specify the following:

```
INC=/usr/local/ssl/include/
LIB=/usr/local/ssl/lib/

all:
gcc -I$(INC) -L$(LIB) -o enc yourcode.c -lcrypto -ldl
```

4. Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanations of the observations that are interesting or surprising. In your report, you need to answer all the questions listed in this lab.