EECE461/ECEG761 Network Security Systems

Homework 2

Cryptography – Secret Key Encryption

(100 points)

Objective

- Get familiar with the concepts in the secret-key encryption.
- Understand encryption algorithms, encryption modes, initial vector (IV).
- Be able to use tools and write programs to encrypt/decrypt messages.

Lab Environment

In this lab, we will use openssl commands and libraries. We have already installed 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 /home/seed/openssl-1.0.1. To configure and install openssl libraries, go to the openssl-1.0.1 folder and run the following command.

```
% sudo ./config
% sudo make
% sudo 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: many 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 using the following command:

```
% sudo apt-get install bless
```

Task 1: Encryption using different ciphers and 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 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. 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".

Task 2: Encryption Mode -- ECB vs. CBC

The file 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:

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 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.

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:
(1) How much information can you recover by decrypting the corrupted file, if the encryption mode 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.
(2) Please explain the reason.
(3) What are the implications of these differences?

Task 4 : 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/descrypt messages in programs.

OpenSSL provides an API called EVP, which is a high-level interface to cryptographic functions. Although 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 in http://www.openssl.org/docs/crypto/EVP_EncryptInit.html. Please get yourself familiar with this program, and then do the following exercise.

You are given a plaintext and a 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 a English word list from the Internet. We have also linked one on the web pageof this lab. The plaintext and ciphertext is in the following:

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

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
```

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 explanation to the observations that are interesting or surprising. In your report, you need to answer all the questions listed in this lab.

Note: This homework is modified from SEED lab below.