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 a 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 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 commands.

```
You should read the INSTALL file first:

  % 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` (ghex in command line). 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 is not installed in the VM version that you are using. To install it, please open the web browser in your VM, go to the website `http://installion.co.uk/ubuntu/precise/universe/b/bless/install/index.html` to find the instructions.

3 Lab Tasks

For this lab, please first create a directory *CryptoLabs* under your home directory. You will put all files related to the three crypto labs in this directory. Please download the necessary files, including *plain.txt*, `pic_original.bmp` etc. from Canvas into this *CryptoLabs* directory.

3.1 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 2 different ciphers and two different modes. 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)
```

3.2 Task 2: 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:

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.

3.3 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 is ECB or CBC, 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 why. (3) What are the implication of these differences?

3.4 Task 4 : Padding

For block ciphers, when the size of the plaintex is not the multiple of the block size, padding may be required. In this task, we will study the padding schemes. The `openssl` manual says that `openssl` uses PKCS5 standard for its padding. The details can be found in `https://tools.ietf.org/html/rfc2898`. Figure 1 shows the basic rule when the block size is 8 (note: in AES the block size is 16).

```
Concatenate M and a padding string PS to form an encoded
        message EM:

                EM = M || PS ,

        where the padding string PS consists of 8-(||M|| mod 8) octets
        each with value 8-(||M|| mod 8). The padding string PS will
        satisfy one of the following statements:

                PS = 01, if ||M|| mod 8 = 7 ;
                PS = 02 02, if ||M|| mod 8 = 6 ;
                ...
                PS = 08 08 08 08 08 08 08 08, if ||M|| mod 8 = 0.

                The length in octets of the encoded message will be a multiple
        of eight and it will be possible to recover the message M
        unambiguously from the encoded message. (This padding rule is
        taken from RFC 1423 [3].)
```

Figure 1: PKCS#5 Padding Mechanism. $||M||$ means the length of the message M.

Please do the following exercises:

1. 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 and CBC modes to encrypt a file, respectively. Please report which modes have paddings and which ones do not. For those that do not need paddings, please explain why.

3.5 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/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 (IDEA encryption) is given in `https:`

//www.openssl.org/docs/man1.0.2/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 web page of this lab (we also provide it in Canvas). 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 plaintex 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. To avoid the above complication, you may directly hardcode the plaintext message in your program.

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. Specifically, the header files for this lab include

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>
```

You also need to tell your compiler where those files are. In your Makefile, you may want to specify the following (where yourcode.c is your c file):

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

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

Note 4: It is somewhat tricky to read line by line the key from the words.txt file. Some words contain 16 or more characters, so when you define a data type for your key, make sure it will not cause buffer overflow. For example, you may either define a larger array for key (e.g., instead of 16 bytes, try 32 bytes) when you use fscanf to read line by line, or make sure every time it will only read at most 16 bytes when you use fgets.

Note 5: Pay attention to format issues. The given ciphertext is in hex format, not a string.

Note 6: The sample code in the website is on the IDEA encryption algorithm, in the EVP_EncryptInit_ex(&ctx, EVP_idea_cbc(), NULL, key, iv) function, you will need to replace EVP_idea_cbc() with EVP_aes_128_cbc();

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