CITS 3004
Cybersecurity

Lab 2: Secret Key Encryption

Task 1: Classical Ciphers

Written by Jin Hong

1A) CAESAR CIPHER (SHIFT CIPHER)

Write a Caesar cipher program that takes two inputs, message `M` and key `K` ($1 < K < 26$), which outputs a ciphertext `C` which shifts the alphabet K places from its original place (e.g., `K` = 3, then a -> d, b -> e, etc).

You have intercepted a message which is known to be encrypted using the Caesar cipher. Decrypt it.

```
Ihwy ojih u ncgy nbyly qum u ahigy hugyx Jynyl. By qum hin u vlcabn
ahigy von by ehyq nbun ihy xus by qcff vywigy nby alyunymn ahigy ypyl
ehiqh. Ohzilnohunyfs, by qum zuwyx qcnb u alyun xuhayl qbyh bcm zlcyhx
Viv xywcxyx nbun by qcff vy nby vymn ahigy ypyl ehiqh. Jynyl xcx hin
quhn ni nolh uauchmn bcm zlcyhx Viv von by qum fyzn qcnb hi wbicwy von
ni ai uauchmn bcg.
```

1B) MONOALPHABETIC CIPHER

An incomplete Monoalphabetic cipher program `monoalpha.py` (written in Python3) is provided, which uses a substitution table using alphanumeric characters including both upper and lower cases (i.e., the table consists of '`abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789`'). Currently, it will randomly generate the substitution order using the shuffle method. Complete the function `key_monoalpha_cipher(key)` such that the user can provide the secret key to use. The order of the substitution table is described in the lectures. For example, if the secret key is 'pineapple', the table is `'pinealmoqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789bcdfghjk'`

1C) RAIL FENCE CIPHER

A Rail Fence cipher `railfence.py` is provided for you. Compare the ciphertexts created using the monoalphabetic cipher when the same plaintext is encrypted.

1D) VIGENERE CIPHER

We have intercepted a ciphertext “ZWHIKVYIGB TEL XXEG WHIK BTI AWHIKKDEYB”. We also figured that a plaintext phrase “OVER” is encrypted to “WHIK”. What other information is needed to decrypt the ciphertext and retrieve the plaintext?

It is said that “Z**WHIK**VYIGB TEL XXEG **WHIK** BTI A**WHIK**KDEYB” the highlighted “WHIK” are correct places for the plaintext phrase “OVER”. Retrieve the key length, and find the key. You are welcome to work with your peers to solve this.

Answer Q1 ~ Q7 on LMS

Task 2: Secret Key Encryption

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.

2A) 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 in the provided VM .

Installing GHex. In this lab, we need to be able to view and modify files of binary format. We have installed in our VM `GHex` a hex editor for GNOME. It allows the user to load data from any file, view and edit it in either hex or asci.

CITS 3004 Cybersecurity

2B) LAB TASKS

2B1) 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 00112233445566778899aabbccddeeff \
   -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`".

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             Key in hex
-iv            initial vector in hex
-[pP]          print the iv/key (then exit if -P)
```

Answer Q8 on LMS

2B2) Encryption Mode – ECB vs. CBC

Download a simple .bmp format image. We would like to encrypt this image, so people without the encryption keys cannot know what is in the image. 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 image as an image, and use an image 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 to directly modify binary files. If you use other image type files, the length of the header varies.

2. Display the encrypted image using any image viewing software. Can you derive any useful information about the original image from the encrypted image? Discuss with your peers/lab instructor

Answer Q9 and Q10 on LMS

2B3) 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 `ghex`.
4. Decrypt the corrupted file (encrypted) using the correct key and IV.

How much information can you recover by decrypting the corrupted file when the encryption mode is ECB, CBC, CFB, or OFB, respectively? Try answering this question before you conduct this task.

Answer Q11 and Q12 on LMS

2B4) 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.

Answer Q13 and Q14 on LMS

2B5) 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 [1]. 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 page of 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
```

An incomplete code can be found on LMS, as well as from [1]. You will have to complete the code given to complete this question. You are welcome to make your own code, as long as it complies with requirements of this question (i.e., using crypto library and its functions).

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 the `ghex` tool to remove the special character.

Note 2: In this task, you are supposed to write your own program to invoke the crypto library. Try to write your own code instead of using 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 prog yourcode.c -lcrypto -ldl
```

Answer Q15 and Q16 on LMS

References

1. http://www.openssl.org/docs/crypto/EVP_EncryptInit.html