Homework 3 – Secret-Key Encryption

1 Overview

The learning objective of this homework is for students to get familiar with the concepts and principles in the secret-key encryption. To achieve the objective, the students will do both theory (paper-and-pencil problems) and practice (programming and lab). After finishing the homework, students should be able to understand better and gain a first-hand experience on encryption algorithms and encryption modes. Moreover, students will be able to use tools and write programs to encrypt/decrypt messages.

2 Paper-and-Pencil Problem [50 pts]

There are a total of 8 paper-and-pencil problems (include one bonus question). [Note: For best results, we recommend you to read the textbook closely before answering these questions (in particular, the KPS textbook [1] could be very helpful).]

- Q1 (5 pts): Random J. has been told to design a scheme to prevent messages from being modified by an intruder. Random J. decided to append to each message a hash of that message. Does this solve the problem? Why?
- Q2 (5 pts): Suppose Alice, Bob, and Carol want to use secret key technology to authenticate each other. If they all used the same secret key K, then Bob could impersonate Carol to Alicia (actually any of three can impersonate the other to the third). Suppose instead that each had their own secret key, so Alice uses K_A, Bob uses K_B, and Carol uses K_C. This means that each one, to prove his/her identity, responds to a challenge with a function of his/her secret key and the challenge. Is this more secure than having them all use the same secret key K? (Hint: what does Alice need to know in order to verify Carol's answer to Alice's challenge?)
- Q3 (5 pts): It is common, for performance reasons, to sign a message digest of a message rather than the message itself. Why is it so important that it be difficult to find two messages with the same message digest?
- Q4 (7 pts): Token cards display a number that changes periodically, perhaps every minute. Each such device has a unique secret key. A human can prove possession of a particular such device by entering the displayed number into a computer system. The computer system knows the secret keys of each authorized device. How would you design such a device?
- Q5 (7 pts): How many DES keys, on the average, encrypt a particular plaintext block to a particular ciphertext block? Please explain.
- Q6 (7 pts): Suppose the DES mangler function mapped every 32-bit value to zero, regardless of the value of its input. What function would DES then compute?
- Q7 (7 pts): The psudo-random stream of blocks generated by 64-bit OFB must eventually repeat (since at most 2^{64} different blocks can be generated). Will $K\{IV\}$ necessarily be the first block to be repeated?
- Q8 (7 pts): Consider the following alternative method of encrypting a message. To encrypt a message, use the algorithm for doing a CBC decrypt. To decrypt a message, use the algorithm for doing a CBC

encrypt. Would this work? What are the security implications of this, if any, as contrasted with the "normal" CBC?

3 Lab and Programming Tasks [50 pts + 15 bonus pts]

There are a total of 5 lab/programming tasks.

3.1 Task 1: Encryption using different ciphers and modes [10 pts]

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`, `-des`, 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/-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 [10 pts]

In this task, we will need a file `pic_original.bmp` which contains a simple picture. (It is available at `http://faculty.cse.tamu.edu/guofei/csce465/pic_original.bmp`.) 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 (i.e., 0x36) 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 `bless` or `ghex` tool (already installed in our VM) to directly modify binary files. We can also use the following commands to get the header from `p1.bmp`, the data from `p2.bmp` (from offset 55 to the end of the file), and then combine the header and data together into a new file.

   ```
   $ head -c 54 p1.bmp > header
   $ tail -c +55 p2.bmp > body
   $ cat header body > new.bmp
   ```

2. Display the encrypted picture using any picture viewing software (e.g., (we have installed an image viewer program called `eog` on our VM). Can you derive any useful information about the original picture from the encrypted picture? Please explain your observations.

Select a picture of your choice, repeat the experiment above, and report your observations.

3.3 Task 3: Encryption Mode – Corrupted Cipher Text [10 pts]

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 `bless`.
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, 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 why. (3) What are the implications of these differences?

3.4 Task 4: Programming using the Crypto Library [20 pts]

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 provided one here (`http://faculty.cse.tamu.edu/guofei/csce465/words.txt`). 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 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. 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
```

3.5 Task 5: Write your own DES encryption code (one round) [Bonus question: 15 pts]

Implement a single round of DES in C(++) or Java *with your own code*. Test with the round keys 0x123456789abc for DES. Measure the execution speed of the implementation (using the time service). (You may need to execute the function more than once to get meaningful results.) The S boxes for DES are provided in the KPS textbook or `http://faculty.cse.tamu.edu/guofei/csce465/sbox.htm` for your convenience.

4 Submission

You need to submit a detailed homework report to describe what you have done and what you have observed (also including screen shots if possible); 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 homework. You need to send all required programs too (together with the report).

References

[1] Charlie Kaufman, Radia Perlman, and Mike Speciner. Network SecurityPrivate Communication in a Public World, 2nd Edition. Prentice Hall, 2002. ISBN 978-0-13-046019-6.