Crypto Coding Project: Encryption and Hashes

1 Overview

The learning objective of this project is for students to get familiar with using secret-key encryption and one-way hash functions in the `openssl` library. After finishing the assignment, in addition to improving your understanding of secret-key encryption and one-way hash function concepts, you should be able to use tools and write programs to generate one-way hash values and encrypt/decrypt messages.

2 Submission Guidelines

This project has a total of 100 points.

You need to submit a report answering the questions associated with the tasks 3.1-3.3 describing your observations for each task. You also need to provide explanations to the observations that are interesting or surprising.

Task 3.1 can be accomplished using OpenSSL commands. Tasks 3.2 and 3.3 require coding. For these you have to write and submit your code along with observations as requested. It is recommended you write them in Python but C or Java is also acceptable. Be sure to include a **README** file on exactly how to compile and run your code. **Submissions must be able to run on OSU FLIP servers**.

Link to access OSU SERVERS
https://it.engineering.oregonstate.edu/book/how-do-i-access-my-engr-account
https://it.engineering.oregonstate.edu/accessing-unix-server-using-putty-ssh
https://it.engineering.oregonstate.edu/accessing-engineering-file-space-using-filezilla-sftp

Your submission should contain the code files, a make/README file describing how to compile/run your code, and a PDF document with observations. Submit these on Canvas as a zip/tar folder.

3 Lab Environment and Tasks

Installing OpenSSL. In this project activity, you will use `openssl` commands and libraries. Please install the latest version of `openssl` compatible with your platform. 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. It is expected that you will accomplish this task successfully on your own, before starting on assignment tasks, following appropriate instructions for your platform. OpenSSL comes pre-installed for many Linux distributions. But you may need to install it for Mac and Windows machines.

It is recommended that you work on OSU FLIP Server if possible as OpenSSL is available and might save you a lot of time. FLIP servers are running an older version of OpenSSL 3.0.7 1 Nov 2022

Be sure to use commands for the version of OpenSSL that you are running. (The commands shown here are samples from an older version of OpenSSL and are provided just as a guide.)

OPENSSL COOKBOOK is a good reference for OpenSSL commands
https://www.feistyduck.com/library/openssl-cookbook/online/

Manual Pages for FLIP version of OpenSSL
https://www.openssl.org/docs/man1.0.2/man1/

One Python Interface to OpenSSL that can be used for is: https://cryptography.io/en/latest/

3.1 [10pts] Observation Task: Encryption Mode – ECB vs. CBC

Encrypt two pictures (convert them to .bmp format first), so people without the encryption keys cannot know what is in the picture. One picture has to be the beaver logo linked here. The 2nd picture can be one of your choices but has to be a natural picture (not animation or computer generated). Please encrypt them 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. You have to set it correctly for the encrypted file to be treated as a legitimate .bmp file. 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? Record both encrypted pictures and the original picture for your report.

What to include in your submission for 3.1:

- Include the original pictures.
- Include the encrypted pictures with CBC mode.
- Include the encrypted pictures with ECB mode.

Example Encryption Command (See the NOTE about `openssl` versions)

```
% openssl enc -aes-128-cbc -e -in plain.bmp -out cipher.cbc.bmp \
                    -K 00112233445566778889aabbccddeeff \
                    -iv 0102030405060708
```

-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. [30pts] Coding Task: Encrypting with OpenSSL

In this task, we will learn how to use OpenSSL's crypto library to encrypt and decrypt messages in programs.

OpenSSL provides a C 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 C code:
https://wiki.openssl.org/index.php/EVP_Authenticated_Encryption_and_Decryption

When working in Python, the pyca/cryptography library provides an interface to the OpenSSL primitives through the “hazmat” namespace and the OpenSSL backend. Similar to the EVP API, the pyca/cryptography library’s hazmat.primitives.ciphers.Cipher provides a way of interacting with the underlying primitives.

Sample Python code is given in
https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/

Sample Java code
https://www.javamex.com/tutorials/cryptography/symmetric.shtml

Please get yourself familiar with one of these two programs and then do the following exercise.

You are given a plaintext and a ciphertext, you know that aes-128-cbc is used to generate the ciphertext from the plaintext, and you also know that the IV is set to 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. We have also linked one here http://www.cis.syr.edu/~wedu/seed/Labs/Crypto/Crypto_Encryption/files/words.txt.

The plaintext and ciphertext are the following:

```
Plaintext (total 21 characters): This is a top secret.
Ciphertext (in hex format):
        8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9
```

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.** Your code will be tested using other input pairs (same format and plaintext length as above) for correctness.

Note 3: To compile your code (this is for C), you may need to include the header files in `openssl`, and link to `openssl` libraries (the paths on flip servers are provided below). 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//include/openssl //or the actual location in your system
LIB=/usr/lib/openssl //or the actual location in your system

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

3.3 [60pts] Coding Task: Weak versus Strong Collision Resistance Property

In this task, we will investigate the difference between hash function's two properties: weak collision resistance property versus strong collision-resistance property. You will use the brute-force method to see how long it takes to break each of these properties. Instead of using OpenSSL's command-line tools, you are required to write your own programs to invoke the message digest functions in OpenSSL's crypto library.

A sample C code is given in https://wiki.openssl.org/index.php/EVP_Message_Digests

Sample Python code is given in https://cryptography.io/en/latest/hazmat/primitives/cryptographic-hashes/

Sample code in Java is given in https://docs.oracle.com/javase/8/docs/api/java/security/MessageDigest.html

Please get familiar with this sample code.

Since most of the hash functions are quite strong against the brute-force attack on those two properties, it will take us years to break them using the brute-force method. To make the task feasible, we reduce the length of the hash value to 24 bits. Once can use any one-way hash function. For consistency, use the first 24 bits of SHA-2 hash value in this task. Essentially, we are using a modified one-way hash function. Please design an experiment to find out the following:

1. [30 pts] Working code that can launch successful weak-collision resistance and strong collision resistance attacks on the modified SHA-2 discussed above

2. [10pts] For the problem above how many trials on average should it take one to break the weak collision resistance property using brute-force method? How many trials did it take for your code to do it? You should repeat your experiment multiple times (15 or more) and report your average number of trials.

3. [10pts] For the problem above how many trials on average should it take one to break the strong collision resistance property using brute-force method? How many trials did it take for your code to do it? You should repeat your experiment multiple times (15 or more) and report your average number of trials.

4. [5pts] Based on your observation, which property is easier to break using the brute-force method?

5. [5pts] Can you explain the difference in your observations?