Introduction à la Securité Informatique 3eme Ingénieur

Tp5

One-Way Hash Function and MAC

Using PGP tool: Kleopatra

Crypto Lab – One-Way Hash Function and MAC

La partie 1 du TP peut être réalisée sous Windows, bien que plus

1 Overview

The learning objective of this lab is for students to get familiar with one-way hash functions and Message Authentication Code (MAC). After finishing the lab, in addition to gaining a deeper undertanding of the concepts, students should be able to use tools and write programs to generate one-way hash value and MAC for a given message.

2 Lab Environment

Installing OpenSSL. In this lab, you will use `openssl` commands and libraries in your Linux 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.
To configure and install `openssl` libraries, run the following commands.

```
You should read the INSTALL file first:

  % ./config
  % make
  % 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. You can use 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: there is another hex editor, called `Bless`, that might be better.

3 Lab Tasks

3.1 Task 1: Generating Message Digest and MAC

In this task, we will play with various one-way hash algorithms. You can use the following `openssl dgst` command to generate the hash value for a file. To see the manuals, you can type `man openssl` and `man dgst`.

```
  % openssl dgst dgsttype filename
```

Please replace the `dgsttype` with a specific one-way hash algorithm, such as `-md5`, `-sha1`, `-sha256`, etc. In this task, you should try at least 3 different algorithms, and describe your observations. You can find the supported one-way hash algorithms by typing `"man openssl"`.

3.2 Task 2: Keyed Hash and HMAC

In this task, we would like to generate a keyed hash (i.e. MAC) for a file. We can use the `-hmac` option (this option is currently undocumented, but it is supported by `openssl`). The following example generates a keyed hash for a file using the HMAC-MD5 algorithm. The string following the `-hmac` option is the key.

```
% openssl dgst -md5 -hmac "abcdefg" filename
```

Please generate a keyed hash using HMAC-MD5, HMAC-SHA256, and HMAC-SHA1 for any file that you choose. Please try several keys with different length. Do we have to use a key with a fixed size in HMAC? If so, what is the key size? If not, why?

3.3 Task 3: The Randomness of One-way Hash

To understand the properties of one-way hash functions, we would like to do the following exercise for MD5 and SHA256:

1. Create a text file of any length.

2. Generate the hash value H_1 for this file using a specific hash algorithm.

3. Flip one bit of the input file. You can achieve this modification using `ghex` or `Bless`.

4. Generate the hash value H_2 for the modified file.

5. Please observe whether H_1 and H_2 are similar or not. Please describe your observations in the lab report. You can write a short program to count how many bits are the same between H_1 and H_2.

3.4 Task 4: One-Way Property versus Collision-Free Property

In this task, we will investigate the difference between hash function's two perperties: one-way property versus collision-free property. We 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 our own C programs to invoke the message digest functions in `openssl`'s crypto library. A sample code can be found from `http://www.openssl.org/docs/crypto/EVP_DigestInit.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. We can use any one-way hash function, but we only use the first 24 bits of the hash value in this task. Namely, we are using a modified one-way hash function. Please design an experiment to find out the following:

1. How many trials it will take you to break the one-way property using the brute-force method? You should repeat your experiment for multiple times, and report your average number of trials.

2. How many trials it will take you to break the collision-free property using the brute-force method? Similarly, you should report the average.

3. Based on your observation, which property is easier to break using the brute-force method?
4. Can you explain the difference in your observation mathematically?

PGP Demonstration

The main function of PGP (Pretty Good Privacy) is to provide secure communication through encryption, authentication, and integrity verification. Use Kleopatra to demonstrate how PGP (Pretty Good Privacy) works,

Requested task: Download the tool from (https://www.gpg4win.org/) for Windows, then:
Create a message, attach a file, sign the message, and send it to your friend.

Questions:
Which key do the sender use to encrypt his message
Which key do the recipient use to verify the signed message
How do PGP verify that your message has not been altered?

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.