IV1013 VT20 – Introduction to Computer Security

Crypto Lab – One-Way Hash Function and MAC

1 Introduction

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 understanding of the concepts, you should be able to use tools and write programs to generate one-way hash value and MAC for a given message. See the Grading section at the end for information about the marking for this lab.

2 Lab Tasks

2.1 Generating Message Digest and MAC

In this task, we will experiment 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
```

Replace the `dgsttype` with a specific one-way hash algorithm, such as `-md5`, `-sha1`, `-sha256`. In this task, you should try these three algorithms (MD5, SHA1 and SHA256). Create a file consisting of your KTH email address in the format `username@kth.se`, with UTF-8 encoding. Generate message digests for this file with all three algorithms.

Question 1. Describe your observations. What differences do you see between the algorithms?

Question 2. Write down the digests generated using the three algorithms.

2.2 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
```

Generate a keyed hash using HMAC-MD5, HMAC-SHA256, and HMAC-SHA1 for the file created in Section 2.1. Try several keys of different length.

Question 3. Do we have to use a key with a fixed size in HMAC? If so, what is the key size? If not, why?

Question 4. Now use the string `IV1013-key` as the secret key and write down the keyed hashes generated using the three algorithms.

2.3 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. Generate the hash value H_1 for the file created in Section 2.1.
2. Flip the first bit of the input file (e.g. **0**1100001 $\rightarrow$ **1**1100001). You can achieve this modification using a binary editor such as `ghex` or `Bless`.
3. Generate the hash value H_2 for the modified file.
4. How similar are H_1 and H_2?

Question 5. Describe your observations. Count how many bits are the same between H_1 and H_2 for MD5 and SHA256 (writing a short program to count the same bits might help you). In the report, specfiy how many bits are the same.

2.4 Collision Resistance

In this task, we will investigate collision resistance of the hash function. The task is limited to weak collisio resistance. We will use the brute-force method to see how long it takes to break this property. Instead of using `openssl`'s command-line tools, you are required to write your own Java program. Get familiar with the provided sample code and feel free to use its parts in your solution. The code uses the `MessageDigest` class from the `java.security` library. You can find more information about this class at the official web documentation:
`https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html`.
Since most of the hash functions are quite strong against the brute-force attack on collision resistance, it could 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. In this task, work with the SHA256 one-way hash function, but use only the first 24 bits of the hash value. Namely, we are using a modified one-way hash function. Please design an experiment to answer the following questions:

Question 6. Investigate how many trials it will take to break the weak collision property using the brute-force method. Below is a list of five messages. For each message, report how many trials it took before you could find a message with the same hash.

- IV1013 security
- Security is fun
- Yes, indeed

- Secure IV1013
- No way

3 Organization

This assignment is done independently.

4 Submission

Upload your solution as a single ZIP archive in canvas. The ZIP archive should contain the following:

- A detailed lab report that describes 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 relevant questions for the problems you have attempted to solve.
- Your Java source code (no compiled files!)
- A file called ”README”, with instructions for how to run your program on the course virtual machine (In Linux shell, no IDE!)
- A file called ”AUTHOR”, with full name and KTH email address of the author.

5 Requirements and Points

For this assignment, you can get a maximum of 100 points. The grading scale is as follows:

- 10 points: your submission compiles correctly, without compilation errors, and your submission meets the requirement above.
- 60 points: In addition to the above, Task 2.1, 2.2, and 2.3 correctly solved.
- 100 points: In addition to the above, Task 2.4 correctly solved.

Note that there can be deduction in points depending on the quality of your solution. If you aim for 100 points, for instance, but we assess that your solution only solves half of the challenges in a satisfactory manner, you might get 80 points.