41900 – Security Fundamentals
One-Way Hash Function – MAC – HMAC

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 understanding 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.

Task 1: Generating Message Authentication Code(MAC)

In this task, we will try various one-way hash algorithms. You should try at least 3 different algorithms and describe your observations.

You can use the **`openssl dgst`** command to generate the hash value for a file. For Example:

- **`openssl dgst -<dgsttype> <filename>`**

  `<dgsttype>`: one-way hash algorithm you want to use, such as

  `-md5`

  `-sha256`

  `<filename>`: name of your file

To see a list of supported algorithms and more options, you can use:

- **`openssl dgst --help`**

Task 2: Generating Hashed Message Authentication Code(HMAC) using keys

In this task, we will 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`**).

You can use the following command to generate a keyed hash for a file using the selected hash algorithm.

- **`openssl dgst -<dgsttype> -hmac <key> <filename>`**

<dgsttype>: one-way hash algorithm you want to use, such as

-md5

-sha256

<key>: the hexadecimal key file used for hashing such as

aabb110022

<filename>: name of your file

Generate a keyed hash using MD5, SHA256, and SHA1 for any file that you choose.

Try several keys with different lengths to observe the difference.

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

Task 3: Testing the Randomness of One-way Hash

To understand the properties of one-way hash functions in a bit more depth, 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 $\mathbf{H_1}$ for this file using a specific hash algorithm.
3. Change one character of the input file.
4. Generate the hash value $\mathbf{H_2}$ for the modified file.
5. Observe whether $\mathbf{H_1}$ and $\mathbf{H_2}$ are similar or not. Optionally, you can write a short program to count how many bits are the same between $\mathbf{H_1}$ and $\mathbf{H_2}$.