CS808: Computer Security Fundamentals

2.2: Article: Cryptographic Hash Functions

Cryptographic Hash Functions

1. Cryptographic Primitives

Before we consider what a cryptographic hash is, it is helpful to note that a cryptographic hash function can be referred to as a cryptographic primitive. A cryptographic primitive can be considered a generic building block of cryptography. Cryptographic primitives are a low level constructs which are used together to build larger cryptographic protocols.

2. Cryptographic Hash Definition

A cryptographic hash function takes data of an arbitrary length, and produces a fixed length string of alphanumeric characters which represents that data. The output can be called a hash value or a digest. I primarily refer to this as a hash value, but you may come across other sources which use the term digest.

There are a number of hash function standards such as SHA-256 and MD5. Note that MD5 is considered insecure for modern practical use, but is commonly used as an example. SHA-256 is commonly used in modern practice.

If calculating the SHA-256 hash of the following string “this is an arbitrary length string” then you should get the hash value:

018D17D6672278F09D99C87B882A4D0AC00CB7322A471BFAA49C6E32C71937E0

Why not try the calculation yourself? One online calculator can be found here https://www.pelock.com/products/hash-calculator though any (correct) implementation of SHA256 will produce the same value.

3. Secure Properties of Cryptographic Hashes

A cryptographic hash function has a number of properties, most notably the following:

It is *deterministic*: The same input using the same hash function always provides the same hash value. This is true for all hash functions, but the following properties are for cryptographic hashes only.

It is *pre-image resistant*: This means that given a random input, it is computationally infeasible to determine the input from the hash value alone. This effectively is a one-way function.

It is *second pre-image resistant*: Given a hash value *h1* it should be computationally infeasible to find a different input message which results in the same hash value *h1*

It is *collision resistant*: It should not be feasible to produce two inputs which have the same hash value as output.

Note the distinction between second pre-image resistance and collision resistance is that in second pre-image resistance you have a given input which it is impossible to find another value which hashes to the same output. For collision resistance, you have no such input and are simply trying to determine two inputs which produce the same hash value.

4. Purpose of Cryptographic Hash Functions

You may be wondering the purpose of cryptographic hash functions. Consider an example as follows, you locate software you wish to download and install from a trusted secure website. On the website you are provided with a link to download the software and a hash value, as well as the specific hash function used to calculate that value. If you then independently calculate the hash value of the file downloaded using that function, and this matches the hash value provided on the website then you have confidence in the integrity of the file. If it were changed, e.g. by an attacker, then the hash value would not match and you would not install the software.

This is the primary use of cryptographic hash functions, we will see how they are used elsewhere as we complete the module.

5. Limitations

As with all things, there are some limitations of cryptographic hash functions you should be aware of.

1. Very small messages or predictable messages cannot be meaningfully hashed. For example, a single bit cannot be meaningfully hashed as any attacker could easily compute the input to result in the hash value.

2. You must trust origin of hash function in order for it to provide integrity, e.g. on a website providing software you would need to be confident of a secure channel which provides the hash value

Last modified: Monday, 3 October 2022, 9:21 AM

◄ 2.1: Video: Cryptography Overview (03:34)

Jump to...

2.3: Video: Substitution Ciphers (07:20) ►