Levi Seibert

00086809

lseiber1@my.athens.edu

ITE 523 – Dr. Yahia Fadlalla

Hashing Algorithms

In its simplest description, a hash is a "code" that is calculated for some piece of data. This code is unpredictable but repeatable, as long as the same data is input. There are many methods of calculating this code, including algorithms like MD5, SHA-1, SHA-2, and SHA-3, all of which will be described shortly. One of the most common applications of hashing is proving the authenticity of some piece of data. A piece of data (available online, for example) will often be provided with a pre-calculated hash value. People who access or download this data can then run it through a hashing algorithm (it must be the same algorithm as was originally used) and check if it matches the provided value. If it does, the data is proven to be authentic; however, if it does not match, then there has been some modification to the data that may indicate some malicious interference or transmission issue. While this is the traditional use of hashing, certain key elements can be added to hashing algorithms to make them especially useful for cryptographic purposes.

Hashing is, at its foundation, a mathematical operation. While the exact process differs based on the algorithm currently in use, all of the major methods have a similar set of characteristics. Hashing is an operation that maps a variable-length (any length) plaintext input to a fixed-length encrypted output. Two important characteristics of hash functions are that they are deterministic, meaning that when they operate on the same piece of data twice, they produce the same output, and they are not invertible, meaning there is no way to discover the input from the output alone. In addition, a hashing algorithm must be efficient and easy to compute for any size block of data. Because they are not reversible, hashing algorithms can be used to build one-way functions.

Hashing is a strong process as long as a trustworthy algorithm is used; however, for use in cryptography, certain additional characteristics have to be present. One of these requirements is that the function must be preimage resistant and 2nd preimage resistant. This means that it is

difficult to find a value that hashes to a given hash value, and it is not possible to find two blocks that generate the same hash value. In addition, cryptographic hashes should exhibit pseudo-randomness so that the outputs are computationally random.

One of the most popular hashing algorithms is MD5. Although it may still be seen in certain applications today, it is no longer considered secure and should not be used for cryptography. MD5 was introduced in 1996 and is still used in applications like file transfers. The very first step of the MD5 algorithm is to pad the message so that it is 64 bits short of being a multiple of 512 bits long (this padding consists of a single 1 followed by however many 0s are needed to meet the bit requirement). The missing 64 bits are used to store the length of the message. For each 512-bit chunk of message, the blocks are split into four mini-chunks, and a predetermined set of operations is performed on each mini-chunk. Some shifting of bits occurs, a complex round function is evaluated, and each of the four mini-chunks are rearranged. One of the key advantages of MD5, and really any good hashing algorithm, is its exemplary implementation of the avalanche property, where a small change in the input will produce an extreme change in the output. Unfortunately, MD5 is susceptible to collision attacks (where the hashes of two different messages are identical) and therefore is not secure enough for modern cryptography.

The Secure Hashing Algorithms (or SHA algorithms) are a collection of multiple iterations of cryptographic hashing functions that have been developed by the National Institute of Standards and Technology and the National Security Agency over the last 20 years. The first version of SHA was SHA-0; however, after a significant cryptographic error was discovered in it, it became obsolete.

SHA-1 corrects the major design flaw in SHA-0. However, it was built on the same foundation as MD5, and as such, it falls victim to several of the same attacks. Despite the fact that vulnerabilities have been found, it is still the most popular version of the SHA family. SHA-1 is used in protocols like SSL and TLS and is even a key component in version control systems like Git. The algorithm for SHA-1 is similar to MD5 and consists of a series of bit shifts, round functions, XOR operations, and sub-block rearranging. Despite the theoretical attacks that have been found for SHA-1, it is technically not broken; however, due to its similarities to SHA-0 and MD5, it is believed to be soon. Therefore, SHA-1 is also best not used for modern cryptography.

SHA-2 is one of today's current hashing standards. It has yet to be broken, but there are fears that one day it will be. SHA-2 actually has multiple variations: SHA-224, SHA-256, SHA-384, and SHA-512, all of which produce digests (outputs) corresponding to the length given in their names, in bits. SHA-224, and -256 use block sizes of 512 bits (like SHA-1 and MD5), while SHA-384 and -512 use 1024-bit block sizes. In addition, these later two versions consist of more steps than -224 or -256 (80 steps vs. 60 steps). The basic methodology of this algorithm consists of a block-chaining-like process where the blocks are split into equal chunks and each chunk is added to the output of the previous chunk's round function. SHA-2, especially the 512-bit digest size, appears to be secure, but it still has similarities with MD5, SHA-0, and SHA-1, and therefore should be used with caution.

SHA-3 was developed after a competition was put on by NIST and NSA in order to find a stronger hashing algorithm that could be used in cryptography. The federal government required that the new standard had to be compatible with SHA-2 so that it could ultimately replace the older algorithm without causing too much overhead in changing data sizes. The evaluation criteria for the Request for Technology considered the algorithms security, cost (in time and money), flexibility, and simplicity. The selected algorithm is the Keccak algorithm, which is relatively new, having become standardized in 2012. The Keccak algorithm is built upon the mathematical concept of sponge functions to hash data. This is a relatively fast process and works well for hardware building. Each round of SHA-3 calls five round functions that consist of XORing, permuting, rotating, and operating on the bits of the data in various ways.

There are many applications of hashes in cryptography, such as producing message authentication codes and digital signatures. A message authentication code, or MAC, is a method used to verify the integrity of a message. A MAC algorithm accepts a secret key and a variable length message and attempts to prove the authenticity of the message by outputting a tag. This tag can be compared with another MAC tag that has been shared with the user to check the authenticity and integrity of the message. A digital signature encrypts the hash value of a message with a user's private key. Anyone with the corresponding public key can prove the integrity of the message.

In addition, there are many uses outside of the cryptographic world. In fact, the most popular use of hashing is proving the authenticity of data, which is not necessarily a

cryptographic concept, although, as is the case with MACs, it can be used in cryptography. One very common application of hashes that everyone uses, but not everyone is aware of is the concept of password hashes. When a password is saved to a device, it is actually the hash of the password that is saved. Whenever the user enters their password, it is hashed and checked against the saved hashes. This is safer and more secure than keeping actual passwords on the computer. Furthermore, certain types of hashing algorithms (like MAC algorithms) have been used to produce pseudo-random number generators, because of the unpredictability of the result of the operation.

Hashing has many applications and can be implemented in several ways; however, the fact that hashes are unique and irreversible are key concepts that allows for several security-related requirements, such as authenticity and integrity.

All information for this report comes from Dr. Adam Lewis' lecture series on hashing algorithms.