Andy Hudson

MTH 312-02

**Securely Condensing Data**

Hash Functions, Message Digests, and Digital Signatures

My main hobby is working on a video game project, which supports multiplayer features. This, of course, requires setup of a client/server network and the use of binary buffers, a method of storing and transferring data over the network. When I started this semester, I seriously ramped up my work on this project (I had slacked for a while) and when I started learning about cryptography in this class, I realized that these buffers I was sending were completely vulnerable to interception/modification. I took a very brief look at the documentation for buffers and found two verification methods: MD5 and SHA-1. Because I will probably need to use at one of these methods in my code, I will explore them further.

When a large amount of data is sent anywhere there is always the risk of that data being compromised and modified for nefarious purposes, especially with our technology increasingly becoming connected to the internet. Therefore, it seems necessary to be able to verify data efficiently, especially sensitive data that must be kept secure (e.g. government use). The message digest achieves this: according to IBM, a message digest is a “fixed size numeric representation of the contents of a message, computed by a hash function,” essentially a smaller numeric summary of a message that is desired to be changed as much as possible by tiny changes in the input – the “Avalanche effect.” While this does create a secure way to easily compare data to check for undesired changes, the method by itself is useless if it is sent along with the actual message; all a bad actor would have to do is make their changes and generate a new digest, making the digest pointless. Therefore, the digest must be encrypted via asymmetric encryption. The Diffie-Hellman protocol that we learned about in class is an example of asymmetric encryption, where encryption is done using a private key and decryption is done with a public key, both from the same source. The combination of a message digest and asymmetric encryption forms a secure digital signature to verify the contents of data sent over a network.

To further delve into the specifics, what is SHA-1? SHA-1 (Secure Hash Algorithm) is a hash function used to generate message digests that was issued as a standard by the National Institute of Standards and Technology (a branch of the U.S. Department of Commerce). There are various methods aside from SHA-1 (e.g. SHA-512, SHA-512/256) that are more secure in that they use larger values but explaining SHA-1 is sufficient. SHA-1 uses bit-strings called “words” and pairs of these “words” called integers, as well as bit shifts, bit rotations, and modular arithmetic to form a sequence of functions. Bitwise XOR is among these functions, to directly relate to our class. The message is first “padded” into a workable format during the preprocessing phase, and the functions are then carried out over a “message schedule” of a fixed size of eighty words, resulting in a 160-bit message digest.

As I learned about in my CIS-263 course (Data Structures), a hash map handles collisions in various ways, whether it be using another data structure to hold collisions or assigning them to a different place in the map. However, for SHA-1, a collision implies that two different messages had the same message digest. If there were any collisions found between two distinct messages, the security of SHA-1 would be completely compromised because a message could be severely and tactically modified to the point where it would return a valid digital signature equal to the original message despite being modified. Exactly this happened in 2017; Google announced that it had found a collision in SHA-1, and the government recommended its use discontinued. I find these revelations interesting because the version control system Git uses SHA-1 for checksums (what I assume are the keys for commits). Given that there are possible collisions in SHA-1, this seems to imply that Git is not a perfect implementation and could fail, costing valuable resources to a company that uses it due to version control errors. MD5 is another message-digest algorithm that predated SHA-1, but like the recently broken SHA-1, MD5 is susceptible to collisions and therefore has virtually been removed from use. There do not appear to be any concepts in MD5 that are not also present in SHA-1, so summarizing how it works would be redundant.

Further points of interest may be: cryptanalysis of MD5 and SHA-1, specifically the concept of the birthday attack and hash collisions; more secure alternatives to MD5 and SHA-1; other encoding functions provided for my game (e.g. base64 encoding).

Sources:

<https://csrc.nist.gov/csrc/media/publications/fips/180/4/final/documents/fips180-4-draft-aug2014.pdf>

<https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10510_.htm>

https://tools.ietf.org/html/rfc1321