Analyzing Hashing Algorithms and The Importance of Securely Storing Information

Thomas Neyman
Email: neymthom@isu.edu

***Abstract*—This paper details the history of hashing algorithms and their uses in the field. The importance of using hashes, their functionality, effectiveness, and individual components are also detailed in this paper. The strengths and weaknesses of the various hashing algorithms are discussed. Finally, a hashing algorithm is implemented in Java to help demonstrate it's process and use for modern day password verification. The implementation can be found on my github account: https://github.com/NeymanThomas/Hashing-System**

Keywords: Hash, Cryptography, Algorithm, Encryption, HMAC, FPGA

I. Introduction

Hashing is a cryptographic technique used in security applications where an algorithm modifies a password or sensitive code in an irreversible way to form a long, seemingly random string of characters. Whenever a secure action needs to be authorized, like logging into an account, the information given is hashed then compared to the hash saved in order to confirm whether or not the same information was given.

Cryptography is widely considered the most effective way at keeping data secure while also satisfying the three properties of CIA in Cyber Security. Those being Confidentiality, Integrity, and Availability. Because of this, hashing is still widely used and accepted in the present [5].

II. History

For the earliest years of computer's history, there wasn't a huge concern to securely hide people's passwords on systems. Not many people were using computers and they were still being developed as the years went on. It was not until the year 1974 when 6th edition Unix was the first system to implement a system for more securely storing the plain text passwords [8]. The system was based off the m-209 cypher and was called Crypt.

The algorithm for hashing passwords at this time was less effective due to the constraints of the time. A better algorithm couldn't be implemented because it simply took up too much space and processing power. As time progressed, the techniques used in hashing algorithms became more sophisticated and other aspects were being considered such as pre-image resistance, second pre-image resistance, and collision resistance.

Currently, there are now many different hashing algorithms developed that can be implemented for security purposes. Some of the algorithms include MD5, Sha-2, RIPEMD-160, MySQL323, bscrypt, and scrypt [4]. All of these algorithms have different strengths and weaknesses with some of them becoming obsolete as new security issues are discovered.

III. Importance of Hashes

Hashing gives a more secure and adjustable method for retrieving data than any other cryptographic structure. Its speed is more efficient than searching for lists and arrays and more importantly, hashed information can't be modified, stolen, or jeopardized. The hash itself can be stolen, but it has no use and can't be applied. The process of comparing hashes is efficient and will instantly determine whether the hashes are distinct or not.

IV. How Hashing Work

There are many components that go into a hashing algorithm as well as some traits that are unique only to some algorithms. At a base level however, every hashing algorithm has the same function. Its purpose is to create a seemingly random and obscure string which represents some input value and can't be reversed.

To get hash values of fixed lengths, the input value is separated into blocks. The hashing function will take these different blocks of fixed length and execute its algorithm on them individually. Processing these blocks one at a time allows the output of the first block to be added to the input of the second block and so on, thus making the final value a combination of all blocks. Simply altering one value in each block will cause the entire hash to change, known as the 'Avalanch Effect' [9].

A. Hashing Tables

A file is needed in order to store the hashes on record. When a password needs to be verified, it is compared to its stored hash using a key. After the key finds the stored hash's location, the two hashes are compared. If they are equal, the password is verified and deemed correct [2].

B. Collisions

Depending on the hashing algorithm, there is a security issues known as collisions that occur. A collision is an instance where two or more passwords will share the same hash, resulting in being able to verify credentials with a completely different password. The MD-5 algorithm is known for being prone to collisons and has fallen off in use because of it[4].

C. Table Chaining

Table chaining is a technique that directly tries to solve the issue of collisions by storing a chain of values at each position in an array instead of a single value [3].

D. Salt & Pepper

A salt refers to a non-secret unique value that is stored somewhere in code or the database then appended to the password before it gets hashed. The salt's purpose is to deter the use of rainbow table attacks where the attacker is able to calculate a table of hashes for given passwords[4, 2].

A pepper is a secret key that is used to convert the hash to a HMAC (Hash-Based Message Authentication Code) [1]. This creates a function that could not be reproduced unless the secret value was known.This adds security if the attacker does not have access to or knows the location of the pepper. If the attacker does discover the pepper's location, the extra layer of security will be lost [4].

V. Effectiveness of Hashes

With many different hashing algorithms to use, it's important to understand the use and ability of each individual one.

A. MD-5

MD-5 is a widely used algorithm but has been used less frequently due to its flawed algorithm causing collisions. However it still functions well against pre-images and second pre-images [4].

B. SHA

Secure Hashing Algorithm is a family of cryptographic hashing algorithms that is maintained by the National Institute of Standards and Technology. Currently there are three algorithms defined: SHA-1, SHA-2, SHA-3 [7].

SHA-1 is similar to the MD-5 algorithm. it uses a 160-bit hash function and had vulnerabilities discovered with it in 2010 which led to the standard being no longer approved for use [7].

SHA-2 is the now used and functional hashing algorithm from NIST which uses a larger bit block size and has less vulnerabilities [7].

SHA-3 is not defined by NIST and is still being developed [7].

C. PBKDF2

This algorithm wasn't originally intended to be used as a password hashing algorithm and was instead used for deriving keys, but because of its slow speed it turned out to be a useful password hasher. It builds upon concepts from SHA-1 and SHA-2 but is much slower and uses an HMAC [1]. The point in having a slower algorithm gets rid of the possibility of brute force attacks and reduces the feasibility of distributed attacks on hashes [4].

D. bcrypt

bcrypt is currently the main in use hashing algorithm for passwords. It derives from the Blowfish block cipher which uses look up tables initiated in memory. Because it needs to set up memory before being used, it can be accomplished on CPU's quite easily but with more difficulty on GPU's [6, 4].

E. scrypt

Using the new concept of instead of adding more and more levels of obscurity and computations to an algorithm to make it overly complex, scrypt uses a process of simply focusing on a single operation that is only possible to solve by a computer. Because it uses a lot of memory and moves the computation to another level, it makes it almost impossible for a GPU or FPGA to attempt to crack it [4].

VI. Hashing System Implementation

In order to better understand the inner workings and functionality of hashing, I implemented a hashing algorithm in Java which stores hashes in a simple JSON file matched with a key (a username). This simplified approach covers all of the functionality that needs to be observed for a hashing algorithm and its applications.

The program allows a user to 'sign up' by inputting a username then a password. They are given the option of wanting to store their password in plain text or as a hash. This option is purely for demonstration purposes to see the flaws of having plain text passwords stored on a file on a database. No real world implementation should ever use plain text passwords as an option.

The program allows the user to retrieve their password from the PlainText.json file by entering their username. The username is used as a key to search the file, then returns the password in plain text to the user. The user may also retrieve their hashed password by the same process. Finally, the user can 'Login' to the program by using their username and password.

A. Entering Information (Plain Text)

When the user is prompted to enter their information as plain text, the program simply stores the username and the attributed password to a JSON file without any security. In figure 2, JohnSmith67 signs up with a password of "GlassHalfEmpty30" and this information is clearly visible and majorly insecure. Any attacker that compromises the database or file where plain text passwords are being stored can clearly see people's information.

B. Entering Information (Hashed)

Now, when the same user JohnSmith67 enters their information again, this time using the hashing algorithm to store their password, we can see that their information is much more secure than before. This is how an attacker who compromised the file would see his information. Notice that the hashes would be of no use and provide no information that can be used to compromise user's accounts.

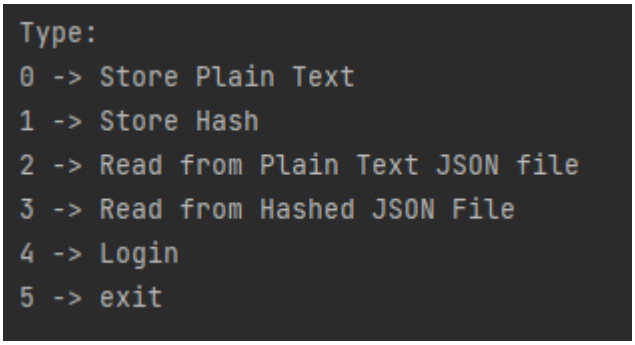

Fig. 1: Menu of the hashing application

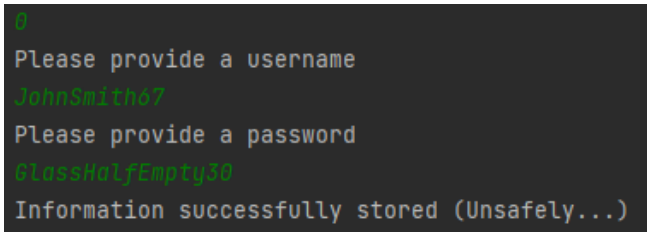

Fig. 2: The credentials provided when storing in plain text

C. Algorithm Implementation

The algorithm used is based off the SHA-256 hashing algorithm and uses many separate steps in order to create a unique hash value for every individual input.

First, constants are created to be used in the algorithm. These constants ensure that the algorithm runs the same way each time. These constants represent the first 32 bits of the fractional parts of the square roots of the first 8 prime numbers. Next, an array of 64 32-bit numbers is also created to be used when the hash is separated into blocks [6].

The input from the user is first converted into a binary sequence with a single 1 being appended on the end. This serves as our salt. Next, 0's are padded onto the binary number until the sequence becomes a multiple of 512. Afterwards, the big endian value of the original input string is used to replace the last 8 bits of the sequence.

Now separating the block in 32-bit arrays, create a total of 64 32-bit array blocks to be iterated over. After each block becomes modified, its output helps modify and shift the bits of the next block and so on. At the end of the iteration, 64 32-bit words will have been created.

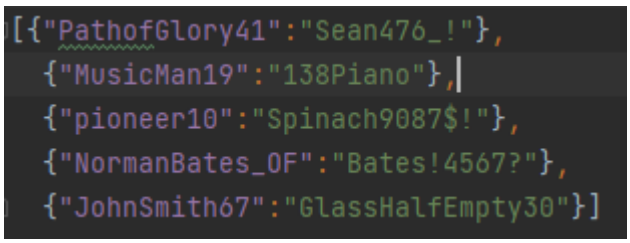

Fig. 3: How the plain text info appears for anyone with access

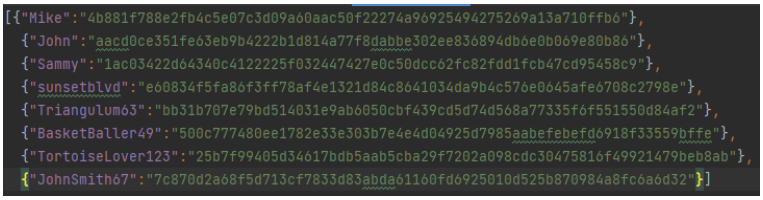

Fig. 4: Hashes of passwords stored in the file

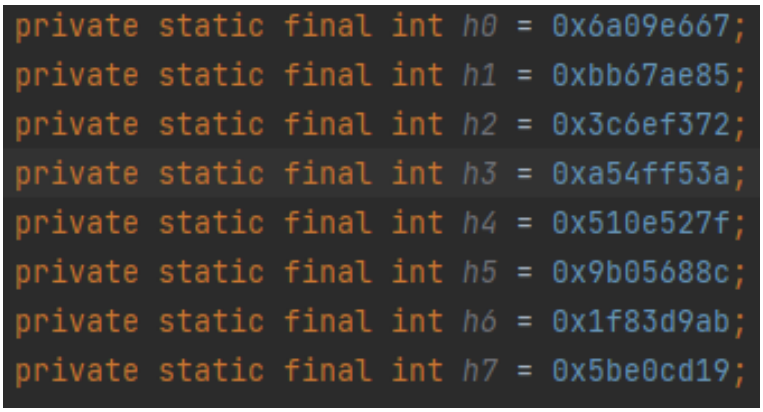

Fig. 5: Predefined constants used to control algorithm

Next a simple compression algorithm using the 8 predefined constants from before is called to take the 64 values and reduce them to smaller strings. Then finally, the string is concatenated to a string and stored.

VII. Conclusion

Through the use of this hashing application, it's easy to see the use and versatility of using cryptography in order to create secure systems. Hashing algorithms are fast, reliable, and offer a great deal of functionality for verifying sensitive information that is too risky to be stored in plain text in a file or on a database.

In this paper I have hopefully demonstrated how hashing systems function and are used in order to maintain secure systems. The basic process for an algorithm has been implemented and demonstrated how hashes were handled and stored compared to vulnerable plain text.

References

[1] H. Krawczyk, M. Bellare, and R. Canetti, "RFC2104," Document search and retrieval page, Feb-1997. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2104. [Accessed: 11-Dec-2021].

[2] L. Constantin, "What is hashing: How this cryptographic process protects passwords," CSO Online, 13-Jan-2021. [Online]. Available: https://www.csoonline.com/article/3602698/hashing-explained-why-its-your-best-bet-to-protect-stored-passwords.html. [Accessed: 11-Dec-2021].

[3] L. Harding, "What is chaining in hash tables?," Educative, 2018. [Online]. Available:

https://www.educative.io/edpresso/what-is-chaining-in-hash-tables. [Accessed: 11-Dec-2021].

[4] L. Kauffman, “About secure password hashing,” About Secure Password Hashing ” Stack Exchange Security Blog, 13-Sep-2013. [Online]. Available: https://security.blogoverflow.com/2013/09/about-secure-password-hashing/. [Accessed: 11-Dec-2021].

[5] L. Nweke, “Using the CIA and AAA models to explain cybersecurity ...,” Using the CIA and AAA Models to Explain Cybersecurity Activities, 2017. [Online]. Available: https://pmworldlibrary.net/wp-content/uploads/2017/05/171126-Nweke-Using-CIA-and-AAA-Models-to-explain-Cybersecurity.pdf. [Accessed: 11-Dec-2021].

[6] L. Wagner, “How sha-256 works step-by-step,” Qvault, 20-Oct-2021. [Online]. Available: https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/. [Accessed: 11-Dec-2021].

[7] National Institute of Standards and Technology, “Secure hash standard (SHS) - NIST,” FIPS PUB 180-4, Aug-2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. [Accessed: 11-Dec-2021].

[8] S. T. amp; A. TrustedSec in Password Audits, H. S. A. Rob Simon in Application Security Assessment, P. T. Drew Kirkpatrick in Application Security Assessment, and P. T. A. D. amp; C. TrustedSec in Penetration Testing, “Of history amp; hashes: A brief history of password storage, transmission, amp; cracking,” TrustedSec, 29-Aug-2019. [Online]. Available: https://www.trustedsec.com/blog/passwordstorage/. [Accessed: 11-Dec-2021].

[9] Savvy Security, “Decoded: Examples of how hashing algorithms work,” Savvy Security, 26-Jan-2021. [Online]. Available: https://cheapsslsecurity.com/blog/decoded-examples-of-how-hashing-algorithms-work/. [Accessed: 11-Dec-2021].