A TECHNICAL ANALYSIS OF HASH FUNCTIONS

This document contains a Technical Analysis of various Hash functions including MD5, SHA1, SHA2, and SHA3.

Jordan Patterson

2024

Contents

Introduction of HASH Functions

Hash functions are an important concept in Computer Science and Information Security. Hash Functions are mathematical algorithms that take an input and then produce an output as a fixed-size string of bytes. The output of a hash function is unique to the input that was given. The same input will always produce the same output. Any change to the input will produce a completely different hash value.

Hash functions are designed to be on-way. They're meant to be fast and efficient when it comes to generating a hash value for an input but theoretically impossible to crack when trying to get that original input value when trying to decrypt that hash value.

There are several types of hash functions such as MD5, SHA1, SHA2, and SHA3. Each hash function is unique and serves its own use cases. MD5 and SHA1 are no longer considered secure whereas SHA2 and SHA3 are still recommended for most use cases.

Hash functions form the base of many algorithms and data structures and their importance in Computer Science and Information Security continues to grow.

Comparison of MD5, SHA1, SHA2 and SHA3

MD5:

Message Digest Algorithm 5 was first designed in 1991 by Ronald Rivest and published in 1992 as RFC 1321. The hash value of MD5 is 128-bit. MD5 turns data of any length into a fixed-length output. MD5 is susceptible to collision attacks making it unsuitable for password storage and digital signatures. Its lower computational requirements make it suitable for tasks such as being a used as a checksum to verify data integrity.

The MD5 hash algorithm can be outlined in 5 main steps:

1. Add Padding Bits to the original input so that it is 64 bits short of any multiple of 512.
2. Add Length bits to the end of the padding so that the total number of bits becomes a multiple of 512.
3. Initialize MD buffer to compute the message digest. MD5 requires a 128 bits buffer with a specific initial value. The buffer is then divided into four words (A, B, C, D), each of which represents a separate 32-bit register.
4. Process each data block in multiple rounds. Each block is processed using 4 rounds of 16 operations and adding each output to form the new buffer value.
5. Produce the final 128-bit hash value.

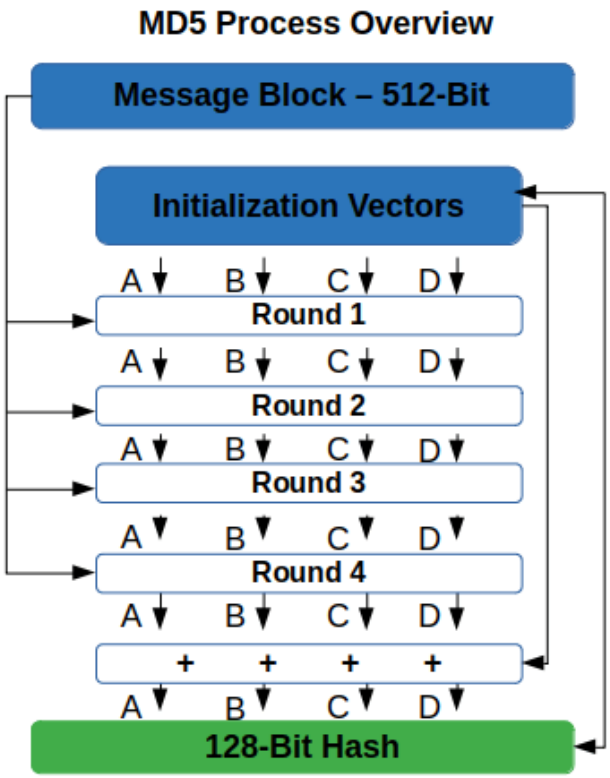

Figure 1. MD5 Hashing Algorithm provided by codesignstore.com

SHA1:

Secure Hash Algorithm 1 was designed by the United States National Security Agency (NSA) and first published in 1995. The length of a SHA1 hash is always 40 characters long. In 2005, several vulnerabilities were found in SHA1, leaving it vulnerable to collision attacks. SHA1 is still sometimes used in services such as TLS, SSL, SSH, and the creation of digital signatures.

The SHA1 hash algorithm can be outlined in 5 main steps:

1. Add Padding Bits to the original input so that it is 64 bits short of any multiple of 512.
2. Add Length bits to the end of the padding so that the total number of bits becomes a multiple of 512.
3. Initialize MD Buffers to compute the message digest. SHA1 requires two buffers and a long sequence of 32-bit words.
4. Process the message in successive 512 bits blocks. Each block goes through a process of expansion and 80 rounds of compression of 20 steps each. The value obtained after each compression is added to the current buffer.
5. Produce the final 160-bit hash value.

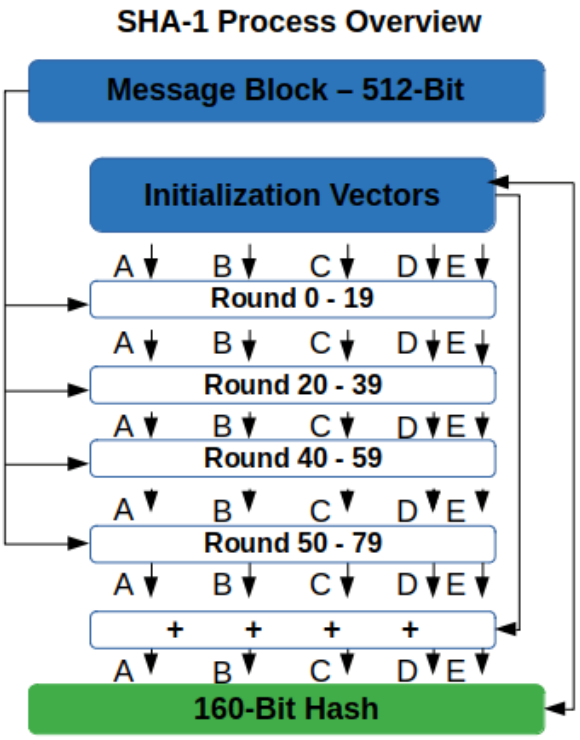

Figure 2. SHA1 Hashing Algorithm provided by codesignstore.com

SHA2:

Secure Hash Algorithm 2 was designed by the United States National Security Agency (NSA) and first published in 2001, 6 years after SHA1. SHA2 turns an input of any length into a fixed-length output. SHA2 is still considered safe to use. SHA2 is used in TLS, IPSec, PGP, S/MIME, SSH, as well as for data authentication and password hashing. SHA2 is a family of hash functions with 6 total hash functions, SHA224, SHA256, SHA384, SHA512, SHA512/224, and SHA512/256.

Each variation of SHA2 (SHA224, SHA256, etc) is computed differently. For this example, we will go over the 5 steps needed to compute SHA2-256:

1. Add Padding Bits to the original input so that it is 64 bits short of any multiple of 512.
2. Add Length bits to the end of the padding so that the total number of bits becomes a multiple of 512.

3. Initialize MD Buffers to compute the message digest. The buffer is represented as 8 32-bit registers.
4. Process the message in successive 512 bits blocks. The message is broken into 512 bits chunks, and each chunk goes through a complex process and 64 rounds of compression. The value obtained after each compression is added to the current hash value.
5. Produce the final 256-bit hash value.

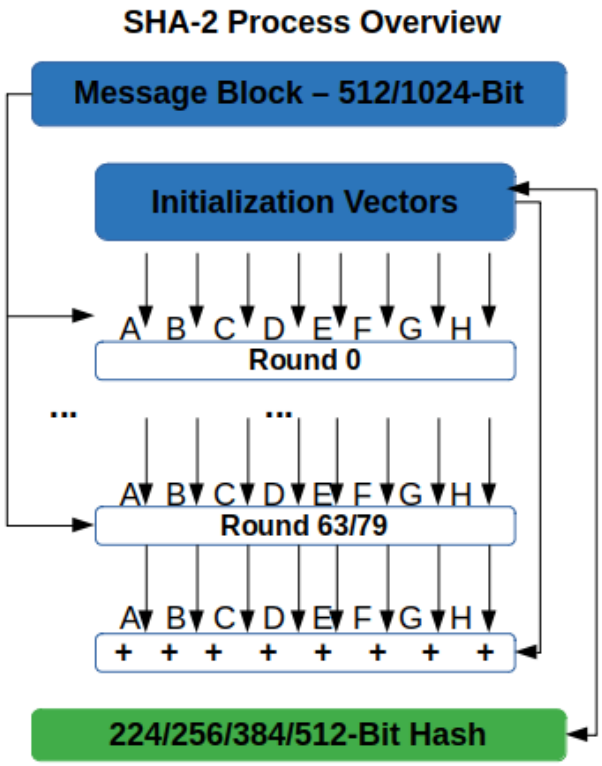

Figure 3. SHA2 Hashing Algorithm provided by codesignstore.com

SHA3:

Secure Hash Algorithm 3 was first published by NIST in 2015, 14 years after SHA2 and 20 years after SHA1. Like the other hash functions, SHA3 takes an input and will produce a fixed length output. SHA3 is considered very secure and safe to use. SHA3 can directly replace

SHA2 in most use cases. Like SHA2, SHA3 is also a family of hash functions containing 6 hash functions. SHA3-224, SHA3-256, Sha3-384, SHA3-512, SHAKE128, and SHAKE256.

Like with SHA2, each variation of SHA3 (SHA224, SHA256, etc) is computed differently. For this example, we will go over the 4 steps needed to compute SHA3-224:

1. Add padding bits to the original message so that the total length is an exact multiple of the rate of the corresponding hash function. For SHA3-224, it must be a multiple of 1152 bits (144 bytes).
2. The padded message is partitioned into fixed size blocks. Then each block goes through a series of permutation rounds of five operations a total of 24 times. At the end, we get an internal state size of 1600 bits.
3. The message is then extracted. The 1600 bits obtained with the absorption operation is segregated on the basis of the related rate and capacity.
4. Produce the final hash value. The first 224 bits are extracted from the 1152 bits. The extracted value of 224 bits is the hash digest of the whole message.

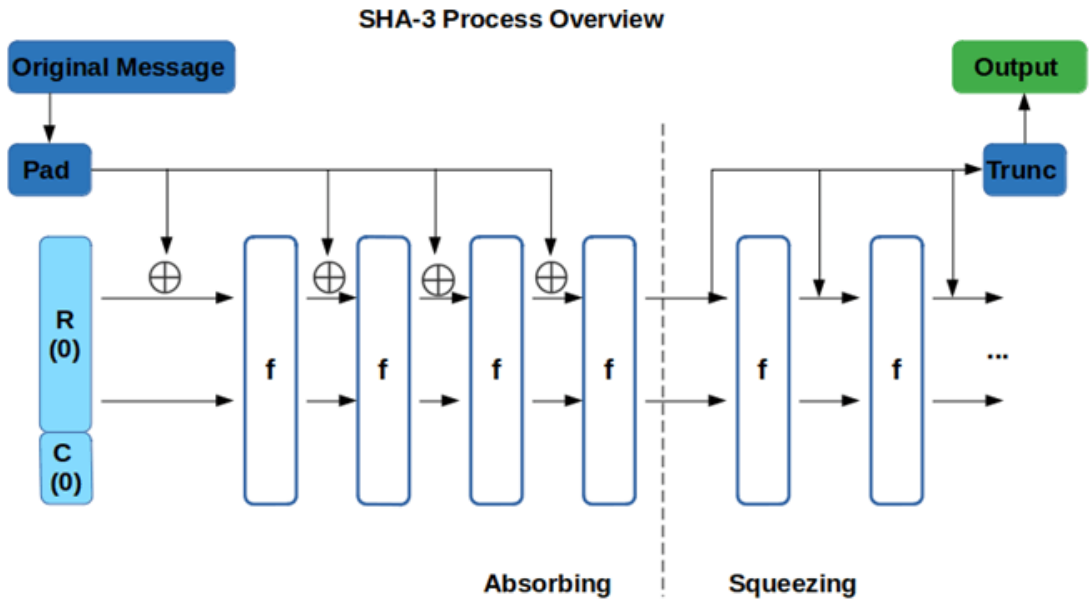

Figure 4. SHA2 Hashing Algorithm provided by codesignstore.com

Hash Algorithm comparison chart:

	MD5	SHA1	SHA2	SHA3
First Published	1992	1995	2001	2015
Block Size	512 bits	512 bits	512/1024 bits	1152/1088/832/576 bits (this is referred to as a Rate [R] for SHA-3 algorithms)
Hash Output size	128 bits	160 bits	256/512 bits	224/256/384/512 bits
Rounds of operation	64	80 (4 groups of 20 rounds)	64 (for SHA-224/SHA-256) 80 (SHA0384/SHA-512)	24
Collision Level	High	High	Low	Low
Common Weaknesses	Vulnerable to collisions	Vulnerable to collisions	Susceptible to preimage attacks	Susceptible to practical collision and near collision attacks.
Security Level	Low	Low	High	High
Application (Use cases)	Verifying data integrity	HMAC, Verifying data integrity	TLS, SSL, PGP, SSH, S/MIME, Ipsec,	TLS, SSL, PGP, SSH, S/MIME, Ipsec,

			cryptocurrency, digital certificates	cryptocurrency, digital certificates
Should still be used?	No	No	Yes	Yes

Explanation of Collision (with example)

Hash functions are a fundamental concept in computer science and cryptography. They are deterministic, meaning that the same input will always produce the same output. This property is essential for many applications, such as data retrieval, password storage, and digital signatures.

However, since the number of possible inputs is greater than the number of possible outputs for most hash functions, collisions can occur. A hash collision is when two different inputs produce the same output hash. This is a rare occurrence but, it is still a possibility and can lead to significant challenges.

Hash Functions and Their Deterministic Nature

Before understanding what a hash collision is you need to understand what a hash function is. A hash function is a process that takes an input and returns a fixed-size string of bytes. The output is typically a ‘digest’ that is unique to each unique input. A slight change in the

input will produce such a drastic change in the output that the new hash value appears uncorrelated with the old hash value.

One of the most important properties of a good hash function is that it is deterministic. This means that the same input will always produce the same output. If you hash the name 'Jacky, you will always get the same hash value. This property is crucial for the function of hash functions in data retrieval and storage.

The Inevitability of Collisions

Despite the deterministic nature of hash functions, collisions are inevitable. This is due to a principle known as the Pigeonhole Principle. The principle states that if you have more pigeons than pigeonholes, and each pigeon must be in a pigeonhole, then at least one pigeonhole must contain more than one pigeon.

In the context of hash functions, the 'pigeons' are the possible inputs and the 'pigeonholes' are the possible outputs. Since the number of possible inputs is greater than the number of possible outputs for most hash functions, there must be at least one output (or 'pigeonhole') that corresponds to more than one input (or 'pigeon'). This is what we call a hash collision.

An Example of a Hash Collision

Here is a simple example of a hash collision using the equation: hash(input_string) = length_of_string(input_string) % 5.

Consider 3 input strings: “apple”, “banana”, and “orange”. The lengths of each string are 5, 6, and 6 respectively. If you apply the hash function to each string, we get the following results:

Hash of “apple”: hash("apple") = 5 % 5 = 0

Hash of “banana”: hash("banana") = 6 % 5 = 1

Hash of “orange”: hash("orange") = 6 % 5 = 1

In this case, there is a collision between “banana” and “orange” because they both produce the same hash value of 1, even though they are different inputs. This collision occurs because the hash function just takes the length of the string modulo 5, and both “banana” and “orange” have the same length modulo 5.

The Implications of Hash Collisions

While hash collisions are a mathematical inevitability, they can pose significant challenges in practice. For instance, in data retrieval, a hash collision can lead to the wrong data being retrieved. In password storage, a hash collision can allow an attacker to gain unauthorized access.

Within cybersecurity, hash collisions can be exploited for malicious purposes. For instance, if a hacker can produce a malicious file that results in the same hash as a legitimate file, they

can potentially bypass security measures. This is why collision resistance is crucial for cryptographic hash functions.

A famous example of a hash collision is the SHAttered attack, where researchers found a collision in the SHA-1 cryptographic hash function. They were able to produce 2 different PDF files that had the same hash value. The discovery led to a decrease in the use of SHA-1 and an increase in the adoption of more secure hash functions such as SHA-256.

So far, SHA-256 has not been compromised to this date. There is still a possibility of it being compromised but it isn't likely right now because of the sheer enormity of processing power it would take to find a collision. To put it into perspective, using brute force, even with all the computational power in the world, it would still take millions of years to find a collision with our current technology.

Hash Collision Conclusion

Hash functions play a crucial role in many areas of computer science and cryptography. While they are deterministic and efficient, the inevitability of hash collisions poses significant challenges. Understanding these challenges and how to mitigate them is essential for maintaining the security and integrity of our data.

Applications of Hashing Algorithms

Hashing Algorithms have various applications in computer science. Some examples of those applications are Password Storage and authentication, Data Compression, Digital Signature Generation, Blockchain, File Comparison, and Fraud Detection to name just a few.

We will now take a deeper look into the application of Password Storage and authentication. Password storage and authentication can be broken down into 4 unique phases.

Password Storage and Authentication

Password Creation:

When the password is created it is passed through a hash function to create the passwords hash.

Storage:

The hashed password is then stored in a database.

Authentication:

When a user attempts to access the system, they enter their password. The password they've entered passes through the same hash function as in step 1 and the hash is produced.

Comparison:

The hash that's just been created is compared with the original stored in the database. If it's a match, the user is authenticated and allowed to log in.

Digital Certificates

Signing digital certificates:

Digital certificates are verified through a process called signing. Signing a certificate involves the server holding the certificate and the CA that is signing the certificate to hash the certificate. The hash is sent along with the certificate by the server to a client who wishes to speak with the server along with the signature of the CA. The CA signature is the certificates hash encrypted with the CA's private key which the client can decrypt with the public key of the CA. The client can then compare the certificate and the decrypted signature to verify they are the same and that the certificate is signed by a trusted CA.

Cryptocurrency

Hashing Algorithms form the basis of Cryptocurrency. There are 3 ways in which hashing algorithms, specifically SHA2-256, play a large role.

Transaction Verification:

All transactions made on the blockchain are hashed using SHA-256. This hash is then included in the block that records the transaction. This will ensure the integrity of the transaction. If the transaction is tampered with, the hash will change, making it easy to detect.

Block Verification:

Each Block in the blockchain contains a hash of all of the transactions that it contains as well as the hash of the previous block in the chain. This creates a link between the blocks which helps with security. If a transaction is altered, the hash of the block and all of the following blocks will change, allowing this tampering to be detected.

Mining/Proof-of-Work:

When mining cryptocurrency your system must solve complex computational problems that involve hashing the data in the block along with a nonce value. You must try and find a hash that is below a certain target value set by the network.

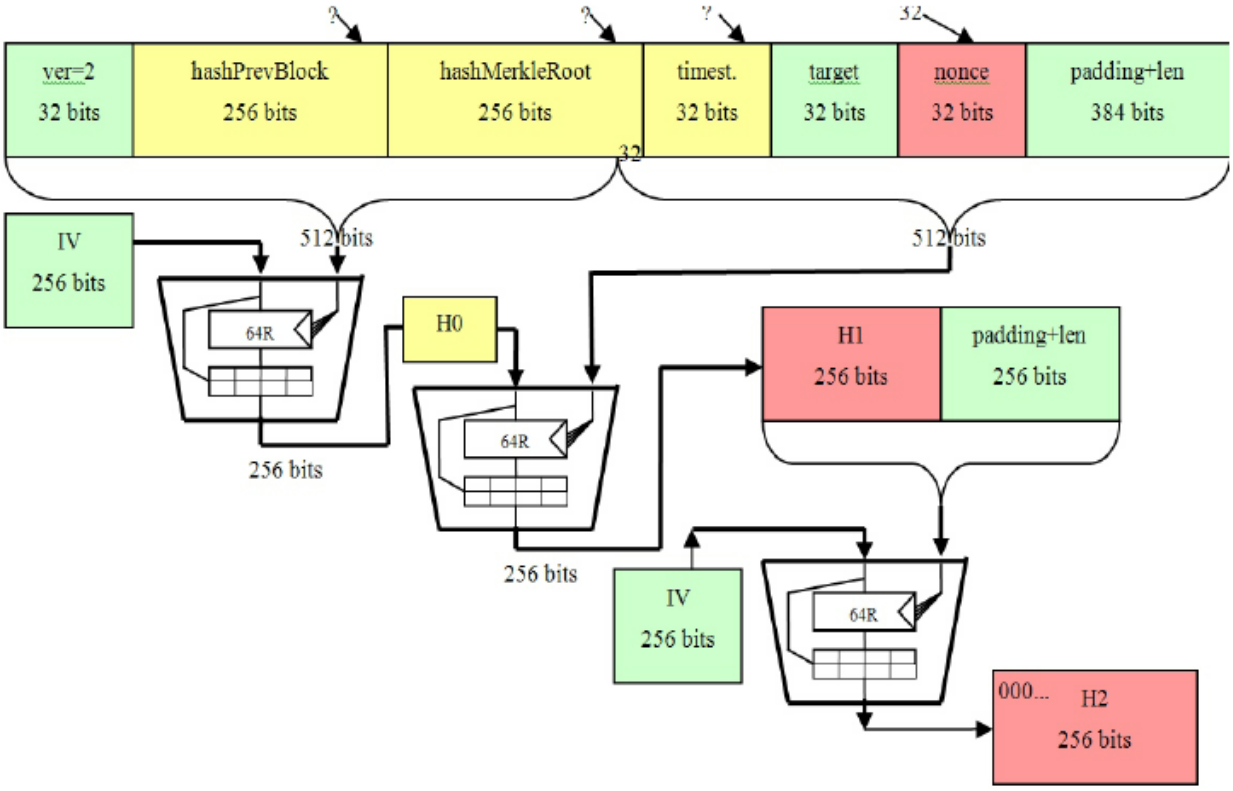

Figure 5. Provided by www.researchgate.net

Summary

Hash functions are important in today's cyber security landscape, they help verify the integrity of files and provide authentication in password management and digital certificates. Hash functions uses math to transform a piece of data into a fixed size message. Hash functions are designed to be quick so they can be used constantly to help provide authentication of file integrity. Hashes are unique depending on the input and if there is more than 1 unique input for any given output it is called a hash collision. Hash collisions are when 2 files or strings have the same hash and the hash algorithms MD5 and SHA1 were discontinued because of the risk of hash collisions. There are 2 main hash functions being used today, SHA2 and SHA3.

References

Hash collision - Wikipedia

A Cryptographic Introduction to Hashing and Hash Collisions - GeeksforGeeks

What are Hash Functions and How to choose a good Hash Function? - GeeksforGeeks

MD5 - Wikipedia

What is MD5 (MD5 Message-Digest Algorithm)? (techtarget.com)

What is MD5 and how is it used? | Comparitech

SHA-1 - Wikipedia

SHA-2 - Wikipedia

What is SHA-2 and how does it work? (comparitech.com)

SHA2 vs SHA3 - A detailed comparison with Examples (debugpointer.com)

SHA-3 - Wikipedia

Hashing Techniques for Password Storage | Okta Developer

Hash Algorithm Comparison: MD5, SHA-1, SHA-2 & SHA-3 (codesigningstore.com)

RFC 1321 - The MD5 Message-Digest Algorithm (ietf.org)

What is Hash Function SHA-256 in Blockchain Technology - Blockchain Knowledge

The process of bitcoin mining according to [5]. | Download Scientific Diagram (researchgate.net)