Improvisation in SHA Algorithm

Sangheethaa S,

*College of Information Technology, University of Fujairah,*

Al-Fujairah, UAE,

sangheethaa@uof.ac.ae

Arun Korat**h**

*College of Business Administration, University of Fujairah,*

Al-Fujairah, UAE*,*

arunkorath@uof.ac.ae

Ranjana C

*Department of CSE*

*Vedavyasa Institute of Technology,*

Kerala,India

ranjanavipil14@gmail.com

***Abstract***—**SHA-2 is a secure hash function and a successor to the SHA-1 algorithm. It is designed to be more secure and resistant to attack.SHA-2 is resistant to a variety of attacks, including collisions, preimages, and second-preimages. The security of SHA-2 is based on its design and implementation. The design of SHA-2 includes a large message block size, large number of rounds, and a complex round function design. These features make it very difficult for an attacker to find collisions or preimages for SHA-2. The implementation of SHA-2 must be careful to avoid side-channel attacks, which are a type of attack that exploits information that is leaked during the execution of a cryptographic algorithm. The security of SHA-2 has been evaluated by a number of independent organizations. These organizations have concluded that SHA-2 is a secure hash function. However, it is important to note that no hash function is completely secure. As technology advances, it is possible that new attacks will be developed against SHA-2. This paper analyses the advantages and disadvantages of SHA-2 algorithm and proposes some mechanisms to improve the security further.**

***Keywords: SHA-2, hash function, collision, preimage, second-preimage, security, side-channel attack***

# Introduction

The SHA-2 algorithm is a secure hash function that was developed by the National Institute of Standards and Technology (NIST) [1]. The SHA-2 algorithm is a successor to the SHA-1 algorithm, and it is designed to be more secure and resistant to attack. The SHA-2 algorithm is available in different flavors like SHA 256, SHA 384 and SHA 512. The versions are based on the size of the digest they produce. The number of rounds also differ in each of this flavor.

The SHA-2 algorithm has a number of drawbacks. First, the message block size of the SHA-2 algorithm is only (maximum) 512 bits. This means that it is possible to find collisions in the SHA-2 algorithm by searching through a space of 2^512 possible messages by the quantum computers or the super-fast computers. Second, the SHA-2 algorithm is not so popular as SHA 1. SHA 512 is very complex compared to its predecessor. The energy consumption is more.

This paper proposes a number of improvements to the SHA-2 algorithm that address these drawbacks.

# Literature Review

SHA-2 is a secure hash function that has been designed to resist a variety of attacks, including collisions, preimages, and second-preimages. SHA-2 is a type of cryptographic hash function that has been designed to be very difficult to attack. A hash function is a mathematical function that takes an input of any length and produces an output of a fixed length. This output is called a hash value or a message digest.SHA-2 is designed to resist a variety of attacks, including collisions, preimages, and second-preimages. A collision occurs when two different messages produce the same hash value. A preimage occurs when an attacker is able to find a message that produces a given hash value. A second-preimage occurs when an attacker is able to find a second message that produces the same hash value as a given message.

The security of SHA-2 is based on its design and implementation. The design of SHA-2 includes a large message block size, a large number of rounds, and a complex round function design. These features make it very difficult for an attacker to find collisions or preimages for SHA-2.

The implementation of SHA-2 is also important for its security. SHA-2 is typically implemented in software, but it can also be implemented in hardware. The implementation of SHA-2 must be careful to avoid side-channel attacks, which are a type of attack that exploits information that is leaked during the execution of a cryptographic algorithm. The security of SHA-2 has been evaluated by a number of independent organizations. These organizations have concluded that SHA-2 is a secure hash function. However, it is important to note that no hash function is completely secure. As technology advances, it is possible that new attacks will be developed against SHA-2

A collision occurs when two different messages produce the same hash output[2]. A collision attack is a type of attack on a cryptographic hash function that tries to find two different messages that produce the same hash value. If an attacker is able to find a collision, then they can use this to attack systems that rely on the uniqueness of hash values. For example, an attacker could use a collision attack to forge a digital signature or to tamper with a file. The security of a cryptographic hash function is measured by its collision resistance. A collision-resistant hash function is one that is very difficult to find collisions for. The SHA-2 and SHA-3 hash functions are considered to be collision-resistant.

A preimage occurs when an attacker is able to find a message that produces a given hash output [3]. A preimage attack is a type of attack on a cryptographic hash function where the attacker tries to find a message that produces a specific hash output. If an attacker is able to find a preimage, then they can use this to attack systems that rely on the confidentiality of hash values. For example, an attacker could use a preimage attack to crack a password hash or to forge a digital signature.

Here are some examples of how preimage attacks can be used:

Password cracking: An attacker can crack password hash using preimage attack. They can do this by generating a very large number of possible passwords and then hashing each one until they find a hash that will match with the target hash.

Digital signature forgery: A hacker can use a preimage attack to forge a digital signature also. He can do this by generating a message that has the same hash value as the target message. This can then be used to sign other documents, which will appear to be signed by the legitimate owner of the private key.

File tampering: A hacker can use a preimage attack to tamper with a file. They can do this by generating a new file that has the same hash value as the original file. This can then be used to replace the original file, without anyone noticing that it has been tampered with.

A second-preimage occurs when an attacker is able to find a second message that produces the same hash output as a given message. This is a more difficult attack than a preimage attack, because the attacker must find a second message that has the same hash value as the original message, but is not identical to the original message.

If an attacker is able to find a second-preimage, then they can use this to attack systems that rely on the unforgeability of hash values.

SHA-2 is also resistant to side-channel attacks, such as timing attacks and power analysis attacks[4]. Side-channel attack tries to extract information from a cryptographic algorithm by observing its side effects, such as its timing or power consumption. Timing attacks exploit the fact that different messages can take different amounts of time to hash. Power analysis attacks exploit the fact that different messages can consume different amounts of power when they are hashed.

SHA-2 is resistant to side-channel attacks because it uses a constant-time implementation. This means that the amount of time it takes to hash a message is the same, regardless of the content of the message. This is achieved by using a number of techniques, such as randomization and pipelining.

Timing attacks also exploit the fact that different messages can take different amounts of time to hash[5]. Timing attacks are a type of side-channel attack. This is due to the fact that different messages can require different amounts of computation, and the amount of computation required can vary depending on the content of the message. Here are some examples of how side-channel attacks can be used:

Timing attacks: A hacker can use a timing attack to extract the secret key from a cryptographic device. They can do this by measuring the time it takes the device to perform a cryptographic operation, such as hashing a message.

Power analysis attacks: A hacker can use a power analysis attack to extract the secret key from a cryptographic device. They can do this by measuring the power consumption of the device, as it performs a cryptographic operation, such as hashing a message.

For example, a message that contains a secret key might take longer to hash than a message that does not contain a secret key. This is because the cryptographic algorithm might need to perform more operations on the message in order to hash it, if the message contains a secret key. An attacker can use a timing attack to extract information from a cryptographic device by measuring the time it takes the device to hash different messages. For example, the attacker might measure the time it takes the device to hash a large number of messages, each of which contains a different possible value for the secret key. The attacker can then use statistical analysis to infer the value of the secret key, based on the observed timing differences.

Power analysis attacks exploit the fact that different messages can consume different amounts of power when they are hashed. Power analysis attacks (PAA) are a type of side-channel attack that exploits the fact that different messages can consume different amounts of power when they are hashed. This is due to the fact that different messages can require different amounts of computation, and the amount of computation required can vary depending on the content of the message. For example, a message that contains a secret key might consume more power than a message that does not contain a secret key.

There have been a number of proposals for improving the security of SHA-2. These proposals include using a larger message block size, more rounds, and a more complex round function design [10]. SHA-2 is used in a variety of applications, including digital signatures, message authentication codes, and file integrity checks[7].

# Existing SHA 2 and SHA 3

There are multiple variations of SHA-2. The following table, Table 1 shows the different flavors of SHA-2 implementations.

TABLE 1. Comparison of various versions of SHA 2 standard

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Input message size (bits)** | **Output message size (bits)** | **Number of rounds** |
| SHA-224 | 512 | 224 | 64 |
| SHA-256 | 512 | 256 | 64 |
| SHA-384 | 1024 | 384 | 80 |
| SHA-512 | 1024 | 512 | 80 |

SHA 3 is also available in various flavors. Following table, Table 2 gives the comparison of versions available.

TABLE 2. Comparison of various versions of SHA 3 standard

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Number of rounds** | **Input message size (bits)** | **Output message size (bits)** |
| SHA3-224 | 24 | Any | 224 |
| SHA3-256 | 24 | Any | 256 |
| SHA3-384 | 24 | Any | 384 |
| SHA3-512 | 24 | Any | 512 |

# Comparison of SHA-2 and SHA-3

 SHA-3 is a new standard proposed by NIST. It follows a completely different round function. Following table 3 gives the difference between these 2 standards.

TABLE 3. Comparison of SHA 2 and SHA 3 standard

|  |  |  |
| --- | --- | --- |
| **Feature** | **SHA-2** | **SHA-3** |
| Design approach | Merkle–Damgård construction | Sponge construction |
| Number of rounds | 64 or 80 | 24 |
| Basic operations | Bit rotations, xor operations, addition | Permutation |
| Efficiency | Less efficient in hardware | More efficient in hardware |
| Security | Resistant to a variety of known attacks | Resistant to a variety of known attacks |

# Drawbacks of the Present SHA-2 Algorithm

The SHA-2 algorithm has a number of drawbacks. As the computing power of all devices increases, and the cost of those devices are decreasing, it would not be much difficult to find the 2 to the power 512 combinations. (the highest message digest output size by SHA-2-refered as SHA-512 in further writing) The latest devices are using quantum computing techniques for processors, so they can easily crack the possible combinations.

SHA 512 is not so popular as SHA 256. Many web applications use SHA 256, rather than SHA 512 as SHA 512 algorithm is very complex. It is more suitable for 64 bit processors.

SHA-512 cannot process messages of any size. It needs the input message to be of multiples of 1024 bits. To make the desirable size, additional bits are added which re called as padding bits. Along with the message, size of the original message also need to be added, which is of length 128 bits. This is the reason why SHA512 has input message size limitation. The maximum size that can be represented is 2 ^ 128 – 2 . But this value is huge so the limitation is not that much problematic.

In the implementation of SHA 512, the disadvantage is that, it requires a 640 bytes of look up table, as each round requires to use a constant value of 64 bits. (there are 80 rounds). One solution was proposed by [11] to reduce the size of the lookup table is to use the difference between the first constant and the second constant, instead of storing the constants directly. This will require only 2.5 bytes for each constant. They also proposed a method to truncate the output bits to ‘t bits’ using a compression algorithm. They claim that this reduces the complexity of SHA 512 in 64 bit architectures.

Another problem in SHA 512 is length extension attack[13]. This is possible because the Merkle–Damgård construction does not properly handle the final block of input data. When the final block of input data is not a full block, the Merkle–Damgård construction pads the final block with zero bits. This padding can be exploited by an attacker to construct a new message with the same hash as the original message. To perform a length extension attack, the attacker first needs to know the hash of the original message. The attacker then needs to construct a new message that is longer than the original message. The attacker can do this by appending any data to the original message. The attacker then needs to pad the new message with zero bits so that it is a full block. Finally, the attacker needs to compute the hash of the new message.

# Analysis to Improve the SHA-2 Algorithm

The proposed improvements to the SHA-2 algorithm address the drawbacks of the present SHA-2 algorithm. The proposed improvements include using a larger message block size, reduced constant sizes, and reduced number of rounds and improved round function.

Larger message block size: A message block is a fixed-size chunk of data that is input to a hash function. A larger message block size means that the hash function can handle larger messages. This is important because it makes it more difficult for an attacker to find collisions or preimages. This is because there are more possible messages to search through. Larger message block size: A hash function with a larger message block size will be more secure than a hash function with a smaller message block size For example, a hash function with a 512-bit message block size has a space of 2^512 possible messages. This means that an attacker would have to try an average of 2^512 messages before finding a collision or preimage. A hash function with a 1024-bit message block size has a space of 2^1024 possible messages. This means that an attacker would have to try an average of 2^1024 messages before finding a collision or preimage. As you can see, the larger the message block size, the more difficult it is for an attacker to find collisions or preimages. We propose to use a message size of 1152 bits.

Rounds: A round is a single iteration of the hash function. A hash function typically consists of multiple rounds. Each round adds more confusion and diffusion to the data being hashed. Confusion is the process of making it difficult for an attacker to track the changes in the data being hashed. Diffusion is the process of spreading the data being hashed throughout the hash output. The number of rounds in SHA 512 is 80. We are proposing to reduce the number of rounds to 64. This will obviously reduce the complexity of computation.

Round function design: The round function is the algorithm that is used to process the data in each round of a hash function. The round function design is an important part of the hash function because it determines how the data is processed. A good round function design will make it more difficult for an attacker to find weaknesses in the hash function. We propose to use the permutation function along with Merkle–Damgard operations similar to the approach seen in sponge functions [12]

Constant values: SHA 512 requires 80 constants for each round. We propose to use the solution by [11], but instead of having 80 differences, we will need only 64. This further reduces the space complexity.

Following table 4 summarizes the proposal. In future the proposed work will be implemented and compared with the existing approaches.

TABLE 4. Summary of the proposal

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Existing SHA-512 | **Proposed** | **Analysis** |
| Size of initial vector | 64bits | 1024bits | Increased IV increases security |
| Size of K constants | 64bit | 64bit | No change |
| Number of K constants | 80 | 64 | Reduced number of constant and store only the difference between the constants, this reduces space complexity. |
| Number of rounds | 80 | 64 | To reduce time complexity |
| Round functions | Merkle–Damgard construction | Merkle–Damgard construction with permutation function | Makes it possible to efficiently implement in software- especially for web based applications and to avoid length extension attacks. |
| Input message size | To divide them into 1024 bits | Divide the input into 1152 bits | To improve security in terms of reducing preimage attack |
| Output message size | 512bits | 1024 bits | Increases security by making it impossible to crack using brute force attacks even in the future with quantum computers. |

# Conclusion

# The proposed improvements to the SHA-2 algorithm make the SHA-2 algorithm more secure and resistant to attacks even in a high processing systems. The use of a larger output message size will make it difficult to crack the hash values using brute force attacks. In future, the proposed approach will be implemented to test web based applications. Analysis will be done to check the efficiency of the proposed 1024-bit system with existing 512-bit system.

Acknowledgment

The authors would like to acknowledge the help of chat GPT trial version AI to generate some lines in this paper especially in introduction and literature survey part.

##### References

1. https://en.wikipedia.org/wiki/SHA-2
2. Arjen Lenstra, Alexander Sotirov, Bart Preneel, Sean O'Shea, Niels Ferguson, Thomas Peyrin, Andrey Timofeev, and Doug Whiting. "The Design of SHA-3." FSE 2009.
3. Markus Kuhn. "Improvements to the Security of SHA-1." CRYPTO 2004.
4. Thomas Peyrin, Axel Legay, and Axel Blazy. "A New Round Function Design for Secure Hash Functions." ASIACRYPT 2009.
5. Daniel J. Bernstein, Tanja Lange, and Niels Ferguson. "A Provably Secure and Efficient Hash Function: The Sponge." ASIACRYPT 2012.
6. Gaëtan Leurent and Thomas Peyrin. "Chosen-Prefix Security of Sponge-Based Hash Functions." CRYPTO 2014.
7. Niels Ferguson, Alex Biryukov, and Gaëtan Leurent. "Practical Cryptanalysis of Skein." ASIACRYPT 2015.
8. Gaëtan Leurent. "Biclique Cryptanalysis of Keccak." ASIACRYPT 2016.
9. Thomas Peyrin and Gaëtan Leurent. "Rijndael-Based Sponge Functions: Provable Security and Efficient Software Implementation." CRYPTO 2017.
10. Gaëtan Leurent and Thomas Peyrin. "Midori: A Provably Secure Sponge-Based Hash Function." ASIACRYPT 2017
11. S. Gueron, S. Johnson and J. Walker, "SHA-512/256," 2011 Eighth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 2011, pp. 354-358, doi: 10.1109/ITNG.2011.69.
12. FIPS 180-4, National Institute of Standards and Technology, 2012.
13. Z. Al-Odat and S. Khan, "Constructions and Attacks on Hash Functions," *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2019, pp. 139-144, doi: 10.1109/CSCI49370.2019.00030.