# Hash functions and the future of SHA-256

December 2021

## 1 Introduction

Cryptographic hash functions are one way functions that map arbitrarily large data fields to a randomised hexadecimal string of fixed length labelled the 'hash' or 'message digest.' They're mainly used in message and password security and are more interestingly applied to cryptocurrency. Hash functions can be thought of as function whose output is easy to compute, but computationally infeasible to reverse [Merkle (1979)]. Hash function SHA-256 plays an integral role in the security of blockchains in cryptocurrencies like Bitcoin. Section 2 discusses key features of a hash function. Sections 3 reviews some past hash functions and why they are no longer viable. Finally, section 4 introduces the main concepts of quantum computing and how it effects the future of SHA-256 and the Bitcoin network.

## 2 What Makes a Good Hash Function?

There are 4 main factors that determine how well a hash function works. The first is hashing speed. The speed at which the message digest is being created has to be fast enough to keep up with normal computers but not too quick as to favour brute force attacks (where an attacker would try every possible combination).

The second characteristic is randomisation. Hash functions are deterministic, meaning the same message will always produce the same message digest. Altering the original message even only by one character should change the message digest completely. This concept aligns with a well known cryptographic result called the Avalanche Effect, which measures how randomised a message digest becomes after slightly changing the original message. By computing a number called the avalanche coefficient (see [Yang, Chen, Chen, Zhang, and Yung (2019)]), we can quantitatively measure the randomisation of different functions. A low avalanche coefficient ($<0.5$) would mean two similar messages would produce a similar message digest, which could leave cryptanalysts with sufficient information to reverse the function and find the original messages. A high avalanche coefficient ($> 0.5$) is ideal.

The third characteristic is that a hash function should be collision-free. Finding two messages that produce the same hash value should be extremely rare. This can be achieved when creating a hash function by using the Merkle-Damgard construction (1979), which is built upon a special type of cryptographic function called an iterative compression function. This takes two fixed length messages and produced a shorter fixed length digest (see [Tiwari et al. (2017)]). The result of a successful collision attack could mean an third party has created an illusion of trust between the sender and receiver and could cause lots of damgage.

Finally the fourth characteristic are pre-image and second pre-image resistances. Pre-image resistance specifies, given a hash value, that it should be computationally infeasible to find a message that produces the given hash value. Second pre-image resistance is as follows: given a message, it should be infeasible to find a second different message such that both messages have the same hash value. It is also a well known fact that second pre-image resistance implies collision resistance [Rogaway and Shrimpton (2004)].

## 3 MD, RIPEMD and SHA families

The MD (message digest) hash family was created by Ronald Rivest starting in 1989 with MD-2. Intended for RSA security, MD-2 takes a message of arbitrary length which is padded to a multiple of 16 bytes, including a special 16 byte quantity called the checksum, and a 128-byte hash value is produced 16 bytes at a time. Each step uses the value of the previous hash and the current message to create the next value. In 1997, it was found that MD-2's compression function was susceptible to collision attacks [Rogier and Chauvaud (1997)]. Thus MD-2 is "no longer considered a secure one-way hash function" [Muller (2004)]. MD-4 was created in 1990 and was used in Windows login and password authentication. However it was found that MD-4 also wasn't collision-free by Dobbertin in 1996 [Dobbertin (1996)]. So Riverst designed MD-5 in 1991 as an improvement over MD-4. MD-4 and MD-5 had the padding and appending steps in MD-2, but differed in the digest process. In MD-4 each block of data was hashed in three complex sub-steps, but in MD-5, it took four sub-steps, making it slower but more secure [Savage (2003)]. However MD-5 was also susceptible to collision attacks, through a 'modular differential attack', which when applied to MD-4 could find a collision in less than a second (see [Wang and Yu (2005)] for details).

RIPEMD is another family of hash functions derived from MD-4, the first being created in 1992 and others following in 1996. The original RIPEMD function stemmed from the Merkle-Damgard construction and used two parallel chains of MD-4. Collision attacks were found in 2004 [Wang, Feng, Lai, and Yu (2004)] and so strengthened versions were designed. In Bitcoin RIPEMD-160 is used after SHA-256 to shorten Bitcoin addresses.

Finally, the most well known family of hash functions is the SHA (Secure Hashing Algorithm) family, published by the National Institute of Security and Technology (NIST). SHA-1, an extension of MD-5, was created in 1995 by the National Security Agency (NSA) and produces a 160-bit message digest. SHA-2 is a mini family of 2 hash functions, SHA-256 and SHA-512. SHA-256 is well known for is extensive use in Bitcoin's blockchain, due to its strong pre-image and collision resistance. [Witoolkollachit (2016)] calculates the average avalanche coefficient of SHA-256 to be 0.9. Whilst both functions are very similar, using SHA-256 over SHA-512 essentially boils down to costs. SHA-512 is faster and more secure, but costs a lot more to store hash values on hardware.

## 4 Quantum Computing

SHA-256's applications to Bitcoin are extremely important in maintaining security throughout Bitcoin's blockchain. Looking back over its predecessors, its important to ask how quantum computers could potentially invert SHA-256 in the future, and what damage this would cause.

To understand how quantum computers could theoretically break the Bitcoin network, we have to understand what goes on inside classical computers and how the rate at which classical computers are speeding up could be translated into quantum terms. In a computer, a transistor is a very small simple component that processes data by either blocking or opening a gate so information can pass through. According to Moore's Law, the number of transistors in an integrated circuit doubles every year. Essentially it's used to predict the rate of increase of computing power annually. Transistors currently are becoming so small that their size has shrunk down to an atomic scale, where the laws of physics act differently.

Randomly guessing the input of a SHA-256 hash is one in every $2^{256}$ hashes which would take millions of years to brute force a pre-image with todays classical technology. The solution lies with quantum comptuers. The quantum version of a bit, called a qubit, is essentially a string of 1's and 0's, except the 1's and 0's represent two states in a quantum system like the spins of an electron. The difference is that while a string of n bits means we use one of $2^n$ possible configurations when processing information, a qubit can be any proportion of both 1's and 0's simultaneously. So instead of using just one, we are able to use all possible configurations at the same time. This phenomenon is called quantum superposition, and can be explained in more detail in [Li, Long, Bai, Feng, and Zheng (2001)]. Qubits also have a property called quantum entanglement, where each qubit reacts to change in other qubits. This link between all qubits means when we observe a single qubit we can gain information about others. These properties gives quantum computers superior computing power over classical computers, and are utilised extremely well when applied to cryptographic problems.

Currently finding a collision or pre-image attack for SHA-256 is almost impossible. The same goes for brute force. Todays technology cannot keep up with the ideas that we need to invert SHA-256. However there is a very real possibility that in 30-50 years when quantum computers have advanced, attacking SHA-256 with brute force based purely off quantum computing power seems more feasible. Grover's Algorithm (1996) [Lavor, Manssur, and Portugal (2003)] could be utilised as a quantum computing algorithm with a quadratic speedup to aid future brute force attacks.

## 5 Conclusions

SHA-256 stems from the design of the MD-4 function. With strong advancements in quantum computing together with Grover's Algorithm, SHA-256 could potentially be inverted which would compromise the entire Bitcoin network. If such an attack was successful and SHA-256 was broken, attackers could essentially circumvent proof of work and mine Bitcoin whenever they want, collecting all mining rewards. There is also a possibility of bringing Bitcoin transactions to a halt, by suddenly pulling away after continuously mining, which would jack up the difficulty adjustment to a point where all other miners no longer have the computing power to solve the required problems. This could maybe be prevented by shifting the Bitcoin network to the quantum world by using special 'post-quantum' cryptography. Whilst it seems like a long way away, preparations for a replacement hash function should be investigated as to avoid an economic disaster.

## 6 Appendix

With our main goal to describe the application of blockchain technology to the current voting system, I detailed the technical aspects of how blocks were created and sealed, using the "BSJC Proof of Completeness" proposed by Shazhad and Crowcroft in 2019. I struggled to keep my section to below a minute and had to cut out a lot of my notes. I was able to manage my time efficiently and reading articles and published papers had helped me with the flow of research in the main assignment. At first I wasn't keen on voting systems and initially preferred to look into weather systems, but I quickly came around after researching. The presentation had greatly helped me with the general style of the assignment and helped with my speaking skills. Overall I believe the project went smoothly and was a nice change of pace in comparison to other assignments.

## References

Dobbertin, H. (1996). Cryptanalysis of md4. In *International workshop on fast software encryption* (pp. 53–69).

Lavor, C., Manssur, L., & Portugal, R. (2003). Grover's algorithm: Quantum database search. *arXiv preprint quant-ph/0301079*.

Li, S.-S., Long, G.-L., Bai, F.-S., Feng, S.-L., & Zheng, H.-Z. (2001). Quantum computing. *Proceedings of the National Academy of Sciences*, *98*(21), 11847–11848.

Merkle, R. C. (1979). *Secrecy, authentication, and public key systems.* Stanford university.

Muller, F. (2004). The md2 hash function is not one-way. In *International conference on the theory and application of cryptology and information security* (pp. 214–229).

Rogaway, P., & Shrimpton, T. (2004). Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International workshop on fast software encryption* (pp. 371–388).

Rogier, N., & Chauvaud, P. (1997). Md2 is not secure without the checksum byte. *Designs, Codes and Cryptography*, *12*(3), 245–251.

Savage, B. (2003). A guide to hash algorithms. *Global Information Assurance Certification. SANS Institute*, *18*.

Tiwari, H., et al. (2017). Merkle-damgård construction method and alternatives: a review. *Journal of Information and Organizational Sciences*, *41*(2), 283–304.

Wang, X., Feng, D., Lai, X., & Yu, H. (2004). Collisions for hash functions md4, md5, haval-128 and ripemd. *IACR Cryptol. ePrint Arch.*, *2004*, 199.

Wang, X., & Yu, H. (2005). How to break md5 and other hash functions. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 19–35).

Witoolkollachit, P. (2016). The avalanche effect of various hash functions between encrypted raw images versus non-encrypted images: A comparison study. *J. Thai Med. Informatics Assoc*, *1*, 69–82.

Yang, Y., Chen, F., Chen, J., Zhang, Y., & Yung, K. L. (2019). A secure hash function based on feedback iterative structure. *Enterprise Information Systems*, *13*(3), 281–302.