SHA-256 AND BITCOIN

TARANG LUNAWAT

1. Introduction

SHA-2 is a keyless hash function family published by the National Institute of Standards and Technology (NIST) in 2001 as an improvement of its previous SHA-1 and SHA-0 hashing algorithms. SHA-2 or Secure Hash Algorithm 2 is comprised of four hash functions: SHA-224, SHA-256, SHA-384, and SHA-512. These functions all have practical applications. Specifically, SHA-256 has a wide variety of uses, including commitment schemes, tamper resistant encoding and signatures, and Bitcoin. This paper will cover the general structure of hash functions, and then cover in depth the SHA-256 hash function. Finally, it will cover the place of SHA-256 in the context of Bitcoin.

2. General Hash Function Structure

Definition 1. A hash function is a function that maps an input m of arbitrary length to an output hash h of fixed length.

This hash is a 'fingerprint' of sorts for the original input, and can be used to verify the original message.

Definition 2. Two distinct messages m_1 and m_2 are said to be a collision if they result in the same hash.

Definition 3. A hash function is said to be collision-resistant if it is difficult to find any two messages that are a collision.

Since the hash length is much smaller than possible input length, there naturally exist less possible hashes than inputs, and therefore collisions. Collision-resistance is concerned with the difficulty of finding such collisions, not their existence.

Definition 4. A hash function is pre-image resistant if it is difficult to find the input m that corresponds to a given hash value.

There is actually no formal proof of this property in the case of SHA-256, but it is generally accepted to be true, and no one has yet found a way to prove otherwise.

Definition 5. A hash function is second pre-image resistant if it is difficult to find another input m_2 that corresponds to the hash value of m_1.

Collision resistance implies second pre-image resistance, but not pre-image resistance.

Definition 6. A hash function is uniformly distributed if each hash has an equal probability of resulting from any message. Any minor variance in inputs should result in great variance in the resulting hashes.

Date: December 2019.

A hash function is said to be secure if it is collision-resistant, pre-image resistant, and uniformly distributed. In addition, a hash function should be fairly efficient to compute.

Typically, hash functions are compromised of two parts: the compression function and the domain extender. The compression function has two inputs, a key and an input message of specified length. Before being passed into the compression function, the block of data is typically padded, both to add variance and ensure that the message is an integer multiple of the required block size. Then, the compression function hashes the processed block using the key. The domain extender strings together multiple compression functions so that the entire hash function can handle an input of any arbitrary length.

The SHA family uses the Merkle-Damgård scheme to extend the domain of the compression function. The first iteration of the compression function is given the first input message block and an initialization vector (IV) as the key. The output hash of that compression function is then used as the key for the next compression function iteration and next block of input. Figure 1 shows a diagram of this scheme.

Example. Let us define our compression function $\mathrm{cf}(m, h) = h - m$. Obviously, this is a very bad compression function, but it will serve our purposes. With $m = 314156$, a block size of 2, and an initialization vector of 42, the execution of the Merkle-Damgård is as follows. (Assume there is no padding.)

We take our first block, 31, and the IV 42. $\mathrm{cf}(31, 42) = 11$. We take this and our message block for the next iteration. $\mathrm{cf}(41, 11) = -30$, and then $\mathrm{cf}(56, -30) = -86$. Sp, our final hash of m will be -86.

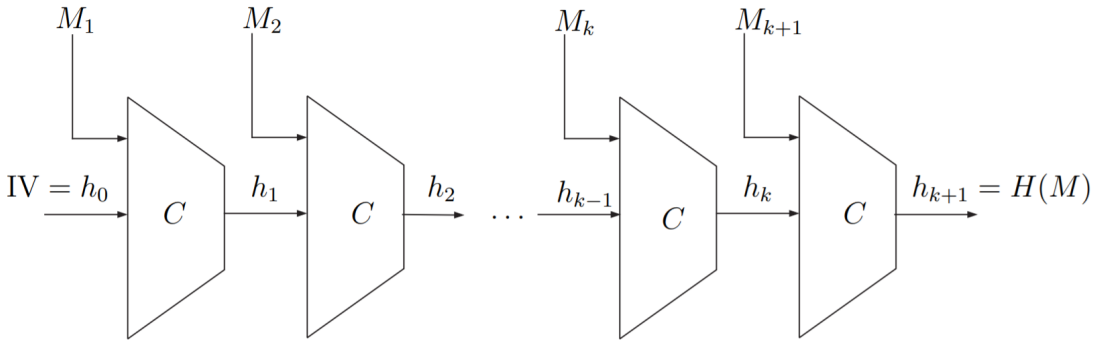

FIGURE 1. [Tel07] The Merkle-Damgård domain extender

3. Message Padding

The hash function is given some message m of length ℓ to operate on. Before iterating through either the domain extender or the actual compression function, m is padded both to create viable inputs for the function and to add variance. We will call the padded message M.

Definition 7. A padding block (PB) is a string of numbers appended onto the end m to create M, ensuring that M's length is an integer multiple of the block size.

SHA-256 splits M into blocks of 512 bits, therefore the length of M must be a multiple of this number. According to NIST specifications, PB start a 1. This one is specifically added instead of just zeroes to add variance that was otherwise lacking. Then, j zeroes are added, such that j is the lowest integer that satisfies $\ell + 1 + j \equiv 448 \pmod{512}$. Finally, the 64-bit representation of ℓ in binary is appended to the end of m, resulting in a total message length that is a multiple of 512. This padding specification constrains ℓ such that $\ell \leq 2^{64} - 1$.

Example. Let us take m to be comprised of 320 0's with 2 1's after it. $\ell = 322$, so $j = 125$. The padded message $M =$ 00011100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101000010. (320 0's, 3 1's, 180 0's, and 322 in binary (9 digits).

4. Constants and Function Overview

SHA-256 iterates on block lengths of 512 bits, denoted $M^{(1)}$ through $M^{(N)}$. $M^{(i)}$ is then broken up into 16 32-bit words, denoted as $M_0^{(i)}$ through $M_{15}^{(i)}$. The compression function iterates 64 times for each M^i, and passes the resulting hash into the iteration for the next message block.

There are a total of 6 functions used in SHA-256: $\mathrm{Ch}(x, y, x)$, $\mathrm{Maj}(x, y, z)$, Σ_0, Σ_1, σ_0, and σ_1. These will be defined in the next section.

In addition, SHA-256 has two working variables (also called chaining variables), T_0 and T_1, which store intermediate values, and eight registers, a, b, c, d, e, f, g, and h.

The initial hash value, H^0, consists of 8 32-bit blocks $H_0^{(0)}$ through $H_7^{(0)}$, which are calculated by taking the first 32 bits of the fractional parts of the square roots of the first 8 prime numbers. The full values can be found in the official NIST documentation [NIS15].

Finally, SHA-256 has 64 32-bit word constants, denoted as K_0 through K_{63}, collectively referred to the key (SHA-256 is called keyless because this "key" is public information, not private). As per NIST standard, these values are the first 32 bits of the fractional parts of the cube roots of the first 64 prime numbers. The full values can be found in the official NIST documentation [NIS15].

5. Function Arithmetic

For each block $M^{(i)}$, the SHA-256 function iterates 64 rounds, starting with $t = 0$ and incrementing up to $t = 63$. Each round takes in an input W_t, defined below, and K_t, as well as the 8 registers a through g. At the end of each round, these registers are updated, used to calculate the i^{th} hash value $H^{(i)}$, and passed as the initialization values when the function processes the block $M^{(i+1)}$. It is these eight registers, that when strung together at the end, result in the message's hash.

SHA-256 uses a message schedule W_t of 64 words labeled W_0 to W_{63}. For message blocks $M^{(1)}$, $M^{(2)}$... $M^{(N)}$,

$$W_t = \begin{cases} M_t^{(i)} & 0 \leq t \leq 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & 16 \leq t \leq 63 \end{cases}$$

All addition is performed mod 2^{32}. Each round of the SHA-256 compressor function uses a value W_t as an input.

The other four functions used in SHA-256 are:

$$\begin{aligned}
\mathrm{Ch}(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\
\mathrm{Maj}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\
\Sigma_0 &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22) \\
\Sigma_1 &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25) \\
\sigma_0 &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \ggg 3) \\
\sigma_1 &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \ggg 20)
\end{aligned}$$

If these symbols are unfamiliar, they are bitwise operators. $\oplus$ is XOR, $\wedge$ is AND, $\neg$ is negation, and $\ggg$ is the right bitwise rotation, which is like a bitwise right shift except that those bits that shift outside of range are taken and appended back onto the left of the number, leaving a result that is the same size as the original number.

Example. The triple bitwise right rotation of 101010, 100011 $\ggg 3 = 011100$

The entire SHA-256 function computation is done as follows. For each block $M^{(1)}$ through $M^{(N)}$, W_t is calculated. The eight registers a through g are set equal to $H_0^{(i-1)}$ through $H_7^{(i-1)}$ respectively for the block $M^{(i)}$.

The hash function runs through 64 iterations, each of which calculates T_1 and T_2 as follows:

$$\begin{aligned}
T_0 &= h + \Sigma_1(e) + \mathrm{Ch}(e, f, g) + K_t + W_t \\
T_1 &= \Sigma_0(a) + \mathrm{Maj}(a, b, c)
\end{aligned}$$

All variables are updated:

$$\begin{aligned}
a &= T_0 + T_1 \\
b &= a \\
c &= b \\
d &= c \\
e &= d + T_1 \\
f &= e \\
g &= f \\
h &= g
\end{aligned}$$

Finally, the i^{th} hash value is computed:

$$\begin{aligned}
H_0^{(i)} &= a + H_0^{(i-1)} \\
H_1^{(i)} &= b + H_1^{(i-1)} \\
H_2^{(i)} &= c + H_2^{(i-1)} \\
H_3^{(i)} &= d + H_3^{(i-1)}
\end{aligned}$$

$$H_4^{(i)} = e + H_4^{(i-1)}$$

$$H_5^{(i)} = f + H_5^{(i-1)}$$

$$H_6^{(i)} = g + H_6^{(i-1)}$$

$$H_7^{(i)} = h + H_7^{(i-1)}$$

This process is repeated for every message block through $M^{(N)}$, with H^N being the final hash value. Figure 2 provides a visual interpretation of an inner round of the compression function.

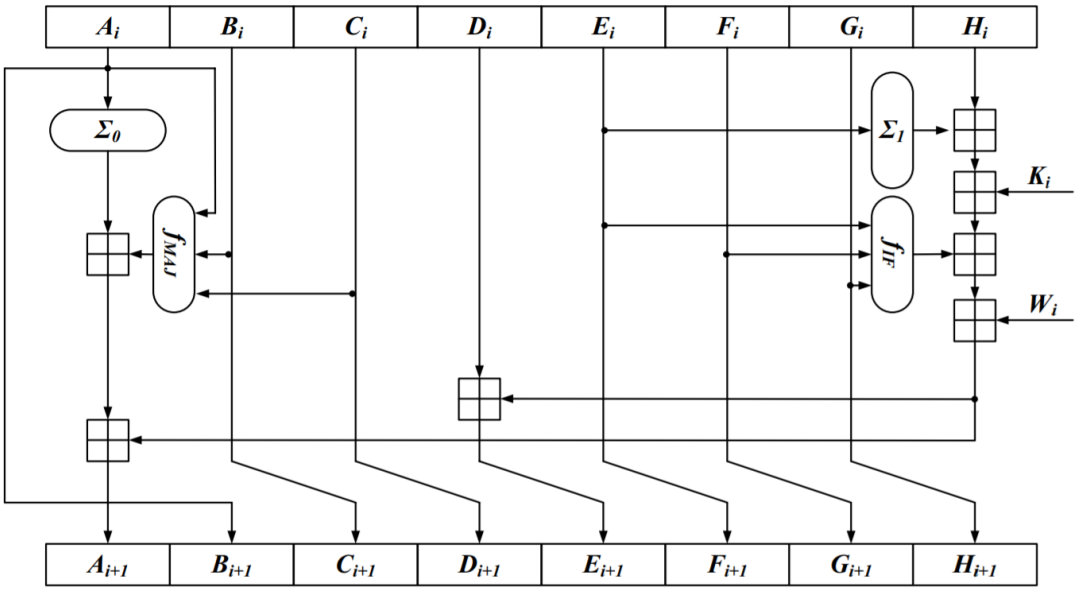

FIGURE 2. [Tel07] Diagram of an inner round of the SHA-256 function. The squares represent addition.

6. In the Context of Bitcoin

Bitcoin is one of the many cryptocurrencies that are in practice today. It, like any cryptocurrency, takes advantage of cryptographic topics to facilitate the exchange of currency. One of the advantages of Bitcoin is that it is decentralized: there is no government to issue the money, nor any bank to manage the accounts and validate the transactions. The "backbone" of Bitcoin is a system of decentralized verification which does not rely on trusting any one party to be honest; it consists of a combination of digital signatures and cryptographic hashes.

Bitcoin is in essence a ledger of all the transactions that have occurred. This record itself the currency. Any new transactions made are simply recorded, without anything being transferred in actuality. In order to avoid being centralized, each user can maintain their own record of the transactions, and each new transaction is publicly broadcasted.

6.1. **Digital Signatures.** Digital signatures are used to verify that transactions are not forged. Each user creates a public and private key (usually, a program does this for them), as well as an address, which is a hash of their public key. Additionally, a user can create as many identities as they'd like, as a way to maintain their anonymity, and prevent observers from linking together all their transactions. When a transaction is broadcasted, the sender of the money adds their digital signature to the bottom of the transaction as a way of signifying that the transaction is what they intended it to be. The signature changes depending on the transaction, so no one can simply copy the signature onto another transaction. In addition, each transaction includes a unique identification number, so identical transactions will still have different signatures. This way, no one can simply execute the same transaction a multiple of times unless the sender of the currency actually intended it. A transaction can only be considered valid if the signature corresponds to the message, and the transaction is done with Bitcoin that the sender has in possession (verifying this requires a history of previous transactions, which is why all are kept).

6.2. **Blockchain.** Bitcoin's ledger is organized into blocks, which each contain a record of multiple transactions. These blocks are chained together to form the complete history of transactions, and are thus called Blockchain.

6.2.1. *Mining.* Each block contains, along with a list of transactions, a hash of the previous block and a nonce, which together are called a Hash Pointer. Anyone can create the next block in a chain, as long as they provide proof-of-work, which is a nonce such that the hash of the previous hash, transactions, and nonce result in a hash that starts with a certain number of 0's and lower than a specified value. This specified value is periodically lowered as the number of miners increases such that each new nonce is predicted to be found in about 10 minutes. Since SHA-256 is uniformly distributed, the only way to find a nonce that achieves this is by brute force. Checking that a nonce is valid, however, is very easy. Once a user finds a nonce, they broadcast their block to other users, who verify it and either add it to their own chain or reject it. A block is considered valid only if it contains a proof-of-work. Users are incentivized to donate CPU to finding nonces by a block reward, which awards whichever user finds and broadcasts the nonce first a certain amount of Bitcoin (this transaction is added to the block). This block reward is an exception to the rules validating transactions. The block reward is not signed, and in addition does not come from existing Bitcoin. All Bitcoin originally come from block rewards, and the reward amount decreases geometrically over time, ensuring that the currency will not be inflated. Searching for nonces to earn Bitcoin is a common practice, and is called mining because it both requires a lot of searching and introduces new Bitcoin into the system.

6.2.2. *Transaction Fees and Non-Mining Users.* Each block is restricted to about 2,400 transactions, which is a slow rate when compared to the rate at which our world operates today. Consequently, there are usually more transactions broadcasted at a time than can be included in a block. To incentivise miners to incorporate a transaction into a block, users leave transaction fees to whoever finds a nonce for the block their transaction is in.

Non-mining users do not have to record every transaction they hear. They can simply listen for new blocks being broadcasted, and record those. In practice, regular users do not have to be fluent in how exactly Blockchain works. They can use programs to handle this all for them.

6.3. **Verification.** Each new block in the chain contains a hash of the previous block in order to preserve the order of the blocks, and therefore verifies the previous block. The standard convention, when presented with two conflicting chains, is to trust the chain that has the most amount of blocks, because it has had the most computational power put into it. Tampering with any one block would alter the hash of that block. Since the hash of that block is used in the next block, it would also alter the hash of the next block and require recomputing the nonce. This process would have to be repeated for every since block in the chain after the tampered block, which would require an unfeasible amount of computational power for one person, since the SHA-256 hash function is hard to invert. The longer chain has a lower probability of being compromised, since it would require more calculation to alter than the shorter chain, and would be harder to forge. If both chains are the same length, users must wait until a block is added to one of them.

6.3.1. *Why Does This Work?* We can explore why this policy works by putting ourselves into the shoes of a dishonest miner. Let us say that Alice has to pay Bob a certain amount of Bitcoin, but she does not want others to know this. She creates a block that includes this transaction, but instead of broadcasting it to everyone, only broadcasts it to Bob. As far as the other miners are concerned, Alice is still in possession of those Bitcoin. Bob will, as a result, be broadcasted two copies of the next block. As per standard, he must keep track of both branches of the chain until one is longer. Since no one else knows about Alice's chain, she must work as fast as all the other miners combined to keep up with their rate of growth. This is, of course, unfeasible unless Alice gets incredibly lucky, as she would need 50% of the computational power that the other miners have combined in order to keep up. Alice will fall behind, and Bob will recognize the block without the transaction as the valid block, therefore rejecting Alice's transaction.

Since it is possible to forge one or two blocks without any issue, a block cannot be trusted until a few blocks are added after it, and it is verified that this is the longest chain. The Bitcoin standard is that a transaction is only considered confirmed when it is part of the longest possible chain, and at least 5 blocks follow it.

6.4. **Other Miscellaneous Comments.** It is possible that two miners discover two different nonces that are valid and broadcast them at the same time. Some people in the network will receive one first, while others will receive the other. In this case, both blocks must be kept, but miners work on the one they received first. Whichever branch receives another block first is considered the official continuation of the chain, and any transactions that were part of the shorter chain once again must wait to be incorporated into a block of the longer chain.

It is also possible that a broadcast of one block reaches most miners, but not all of them. In this case, when a miner that received the block broadcasts the next block to a user that did not, that user will realize that the hash contained in the new block does not correspond to the last block that they have recorded. They can then request the missing block.

Bitcoin's system is based around four of its qualities, which are what make it a compelling system to use: semi-decentralized, publicly verifiable, tamper resistant, and eventually consistent. Bitcoin is semi-decentralized in that it is not maintained by any one entity, but a collaboration of a network of computers. It is publicly verifiable because both the transactions and the hash function are public knowledge. Anyone can create as many identities as they would like, and all the pseudonyms look the same, maintaining anonymity. The tamper

resistant quality of Blockchain was described above, and Bitcoin is eventually consistent in that the longest chain of transactions becomes the main chain.

This paper provides a basic overview of how Bitcoin functions, but not all details. Data efficiency based on Merkle trees are not covered, as are multiple in/out transactions or change. An overview of these is given in the original Bitcoin paper [N+08].

7. Conclusion

SHA-256 was developed as part of SHA-2, which builds on SHA-1 and is considered more secure. However, as the capabilities and speed of computerized calculations increases, SHA-2 will become less secure, and is projected to last less than a decade. Attacks have already been developed that compromise the security of SHA-256, and so SHA-3, the next in the series, is already being developed. It is likely to have an even more computationally complex hash function, and will eventually become the standard, with SHA-2 being abandoned like the previous versions of the family.

References

[N+08] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[NIS15] NIST. *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Secure Hash Standard (SHS)*, August 2015.

[Tel07] Gerard Tell. Cryptography in context. 2007.