Cryptographic Hashing and Application to Blockchain Structures
Draft Summary

Grant Keller

Obscuring information plays a huge roll in computer science and its plethora of uses continues to grow. Hashing is one of those uses. Hashing allows for any quantity of data to be transformed into a string of seemingly random letters and numbers. Hashing is a one-way form of encryption, meaning once something has been hashed you cannot un-hash it. How is this valuable in computer science and how is this the backbone for a 100-billion-dollar industry?

There are many times when developing software where it is useful to protect the original data, but if you still need to manipulate the data what do you do? Hashing allows a nice workaround in a lot of situations, if you need to store passwords without storing the actual password itself. Only save hashed passwords in your data base. Maybe you have a map of information, hashmaps are used to store mapped information in the form of hashes. Block chain is a relatively new concept that took root when Satoshi Nakamoto published a paper on crypto currency. The idea was a peer to peer currency that used a block chain data structure to house transactions. In essence, one uses a chain of blocks of data to store transaction information, each block containing a certain amount of transactions made on the network. In the past, similar block style data structures called linked lists were connected by a pointer that held the address of the next block. Satoshi proposed that the pointer be the results of all the data in the current block hashed. This way, if someone tried to change the data in that block, the hash would change and the block would no longer be connected to the blockchain.

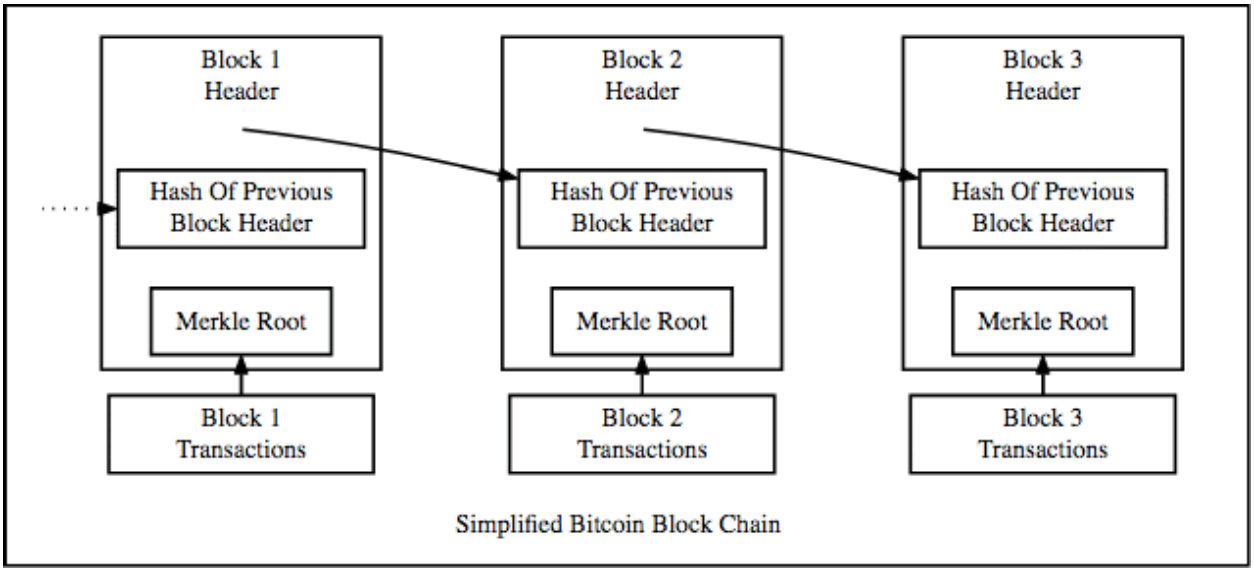

Image from BlockGeeks.com

The math behind Hashing

Sha-1

1- The input is taken in, and is assigned up to 512 bits worth of zeros(64 characters). This is called the padded input. Five more “words” of 512 bits are used(The words are always the same, this guarantees the same result if done multiple times). A ‘word’ is a string of binary numbers that will be used in the computations.
2- The six words are then converted to binary and we perform XOR operations to each word 80 times. An XOR operation is like addition except you never carry over to the next place value but still place a zero in the current place value.
   Example:
   011010
   010111 XOR

   = 001101

3- Circular shifting. With the other five words we shift the bits all to the right and take the bit on the farthest right and bring it to the beginning. The algorithm states that if the word you are working with has a value greater than 32 than right shift. If it is less than 32, left shift.

   Example

   010111 << right circular shift – right most bit got moved to the left most position.

   =101011

4- We shuffle the positions of our words before performing a function on the middle three, this is done 80 times. The function is a little complex and I won’t go into details but I will give a brief example. It is
   I + ((II OR III) AND (II OR IV)) +V + V1 – done 80 times.
   The AND represents a binary operation, It is multiplication of the place value.
   The OR symbol is similar to XOR above except that if any place has a 1 the returned result has a one in that place value.
   The roman numerals represent the position of the six words that we are operating on.
   Example – We will use six bit binary numbers even though in reality each word is much larger.
   Note – addition is done in base 2 and the + sign is not a special operation.
   I + ((II OR III) AND (II OR IV)) +V + V1
   1) 001100+ ((100101 OR 000101) AND (100101 OR 000111)) + 110011 + 011100
   2) 001100+ (100101) AND (100111) + 101111
   3) 001100 + 100111+ 101111
   4) 001100 + 010110

5) 011010

For a more thorough explanation on bitwise operations and the math shown as Summations see the link below.
https://wikimedia.org/api/rest_v1/media/math/render/svg/3d9b60e8c1ae57db132118e72d234993b0e73576

5- The last operation performed before returning the hash is shown as such
Result = I^128 AND II^96 AND III^64 AND IV^32 AND V
And the result is returned as letters and numbers and not as binary.

This is the process the computer goes through when hashing data. It seems complex but the operations are easy for the computer to do. Hashing is special because it is a one way operation that is irreversible and allows for blocks in blockchains to connect to each other in a manner that doesn't allow for the chain to be altered.