<https://blockgeeks.com/guides/what-is-hashing/>

**What Are Addresses on Blockchains?**

In the early days of [Bitcoin](https://blockgeeks.com/guides/what-is-bitcoin-a-step-by-step-guide/), it was possible to send[payments to an IP-address](https://en.bitcoin.it/wiki/IP_transaction) like 104.25.248.32 (which is blockgeeks.com). This was planned to be a convenient method to use Bitcoins without dealing with unhandy public keys and addresses. However, after the Bitcoin developers realized that this way of sending coins could be subject to serious man- in-the-middle-attacks, the option was disabled and did never come back.

**The Public Key: Where the Blockchain Address Generation begins**

After Pay to IP had been abandoned in [Bitcoin](https://blockgeeks.com/guides/what-is-bitcoin-a-step-by-step-guide/), P2PKH became the new standard format for Bitcoin addresses.

It looks like this:

**1K31KZXjcochXpRhjH9g5MxFFTHPi2zEXb**

A standard P2PKH address has something like 34 signs and starts with a 1.

P2PKH is the abbreviation of “Pay To Public Key Hash.” This means that you pay to a hash of a public key.

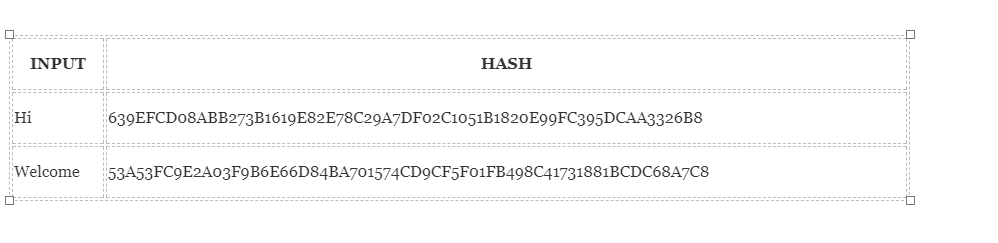
**Examples of cryptographic hash functions**

* MD 5: It produces a 128-bit hash. Collision resistance was broken after ~2^21 hashes.
* SHA 1: Produces a 160-bit hash. Collision resistance broke after ~2^61 hashes.
* SHA 256: Produces a 256-bit hash. This is currently being used by Bitcoin.
* Keccak-256: Produces a 256-bit hash and is currently used by Ethereum.

**So what is hashing?**

In simple terms, hashing means taking an input string of any length and giving out an output of a fixed length.

In the context of cryptocurrencies like Bitcoin, the transactions are taken as an input and run through a hashing algorithm ([Bitcoin uses SHA-256](https://blockgeeks.com/guides/blockchain-address-101/)) which gives an output of a fixed length.



As you can see, in the case of [SHA-256](https://en.wikipedia.org/wiki/SHA-2), no matter how big or small your input is, the output will always have a fixed 256-bits length.

This becomes critical when you are dealing with a huge amount of data and transactions. So basically, instead of remembering the input data which could be huge, you can just remember the hash and keep track.

**Various properties of hashing functions and how they get implemented in the blockchain.**

A cryptographic hash function is a special class of hash functions which has various properties making it ideal for cryptography.

There are certain properties that a cryptographic hash function needs to have in order to be considered secure.

1. **Deterministic -** This means that no matter how many times you parse through a particular input through a hash function you will always get the same result.
2. **Quick Computation -** The hash function should be capable of returning the hash of an input quickly. If the process isn’t fast enough then the system simply won’t be efficient.
3. **Pre-Image Resistance -** What pre-image resistance states is that given H(A) it is infeasible to determine A, where A is the input and H(A) is the output hash.

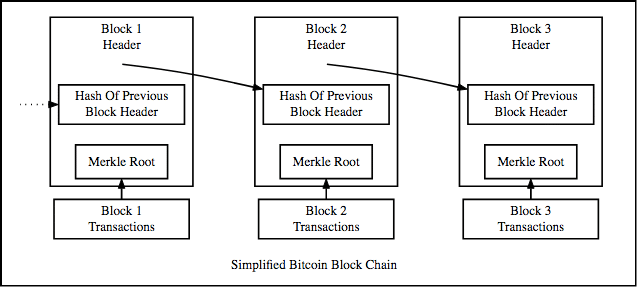
Suppose you are rolling a dice and the output is the hash of the number that comes up from the dice. How will you be able to determine what the original number was? It’s simple all that you have to do is to find out the hashes of all numbers from 1-6 and compare. Since hash functions are deterministic, the hash of a particular input will always be the same, so you can simply compare the hashes and find out the original input.

But this only works when the given amount of data is very less. What happens when you have a huge amount of data? Suppose you are dealing with a 128-bit hash. The only method that you have to find the original input is by using the “[brute-force method](https://en.wikipedia.org/wiki/Brute-force_search)”. Brute-force method basically means that you have to pick up a random input, hash it and then compare the output with the target hash and repeat until you find a match.

So, while it is possible to break pre-image resistance via brute force method, it takes so long that it doesn’t matter.

1. **Small Changes In The Input Changes the Hash-**
2. **Collision Resistant -** Given two different inputs A and B where H(A) and H(B) are their respective hashes, it is infeasible for H(A) to be equal to H(B)

**A block chain is basically a linked list. Let’s see what the blockchain structure looks like:**



The blockchain is a linked list which contains

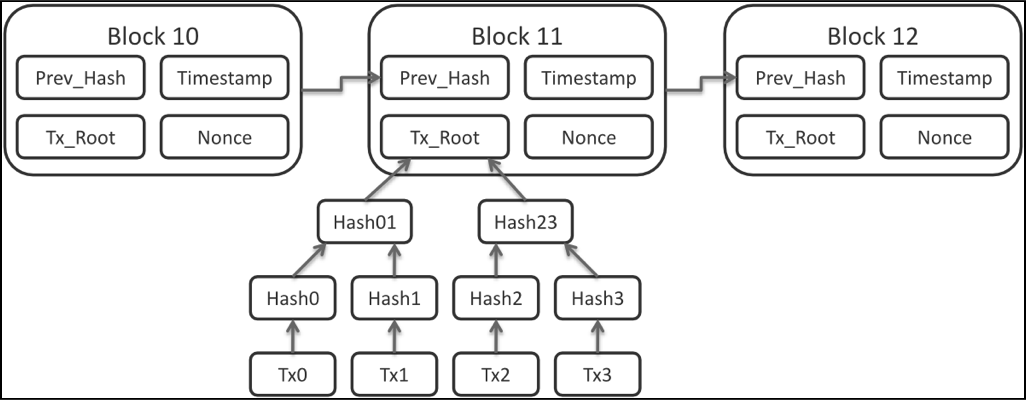
1. data and
2. a hash pointer which points to its previous block,

A hash pointer is similar to a pointer, but instead of just containing the address of the previous block it also contains the hash of the data inside the previous block.

Imagine this for a second, a hacker attacks block 3 and tries to change the data. Because of the properties of hash functions, a slight change in data will change the hash drastically. This means that any slight changes made in block 3, will change the hash which is stored in block 2, now that in turn will change the data and the hash of block 2 which will result in changes in block 1 and so on and so forth. This will completely change the chain, which is impossible.

This is exactly how blockchains attain immutability.

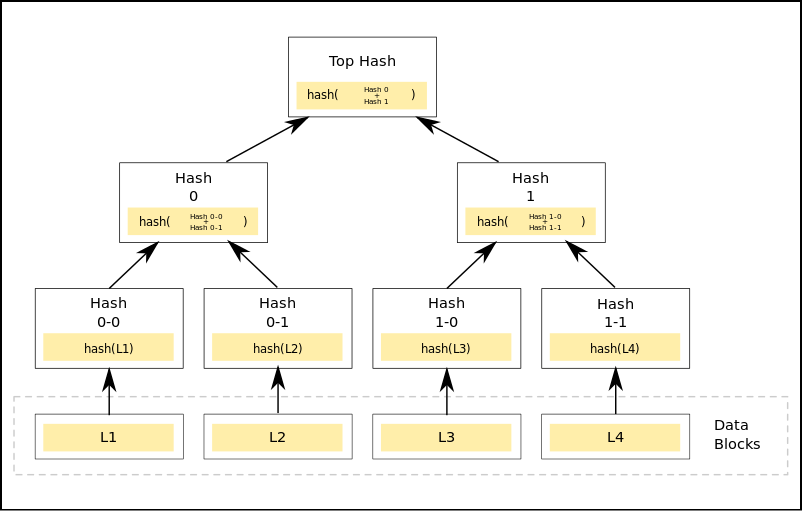
**So what does a block header look like?**



**A block header contains:**

* Version: The block version number.
* Time: the current timestamp.
* The current difficult target. (More on this later).
* Hash of the previous block.
* Nonce (more on this later).
* Hash of the Merkle Root.

**What is a Merkle Tree?**



Merkle trees are used to summarize all the transactions in a block, producing an overall digital fingerprint of the entire set of transactions, providing a very efficient process to verify whether a transaction is included in a block.

In a Merkle tree, each non-leaf node is the hash of the values of their child nodes.

Leaf Node: The leaf nodes are the nodes in the lowest tier of the tree. So wrt the diagram above, the leaf nodes will be L1, L2, L3 and L4.

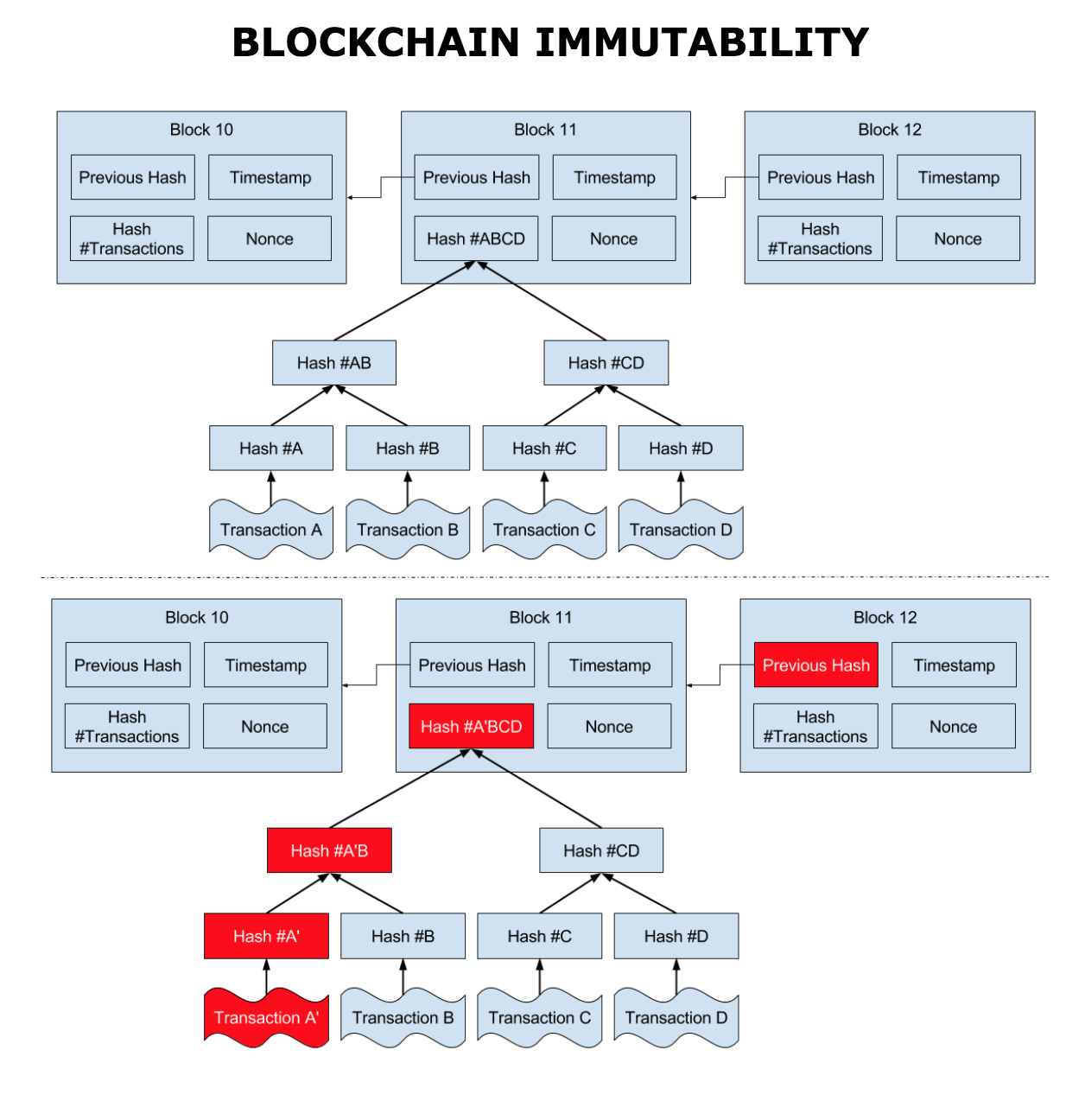
Child Nodes: For a node, the nodes below its tier which are feeding into it are its child nodes. Wrt the diagram, the nodes labeled “Hash 0-0” and “Hash 0-1” are the child nodes of the node labeled “Hash 0”.

Root Node: The single node on the highest tier labeled “Top Hash” is the root node.

**So what does a Merkle Tree have to do with blockchains?**

Each block contains thousands and thousands of transactions. It will be very time inefficient to store all the data inside each block as a series.

If we use a Merkle tree, however, we will greatly cut down the time required to find out whether a particular transaction belongs in that block or not.



**Hashing in mining: The crypto puzzles.**

When we say “mining”, it basically means searching for a new block to be added in the blockchain.

Miners from around the world are constantly working to make sure that the chain keeps on growing.

Earlier it used to be easy for people to mine using just their laptops, but over time, people started forming mining pools to pool in their computer powers and mine more efficiently.

This, however, could have been a problem. There is a cap for each cryptocurrency, eg. for bitcoin, it is just 21 million.

There are only 21 million bitcoins out there. If the miners are allowed to carry on, at this rate, they will fish out all the bitcoins in existence.

On top of that, there needs to be a specific time limit in between the creation of each blocks. For bitcoin, the time limit in between block creation is 10 mins.

If the blocks were allowed to be created faster, it would result in:

* More collisions: More hash functions will be generated which will inevitably cause more collisions.
* More orphaned blocks: If a lot of miners are over mining they will come up with new blocks simultaneously. This will result in or more blocks not getting to be part of the main chain and becoming orphan blocks.

So, in order to restrict block creation, a specific difficulty level is set. Mining is like a game, you solve the puzzle and you get rewards. Setting difficulty makes that puzzle much harder to solve and hence more time-consuming. WRT bitcoins the difficulty target is a 64-character string (which is the same as a SHA-256 output) which begins with a bunch of zeroes. A number of zeroes increases as the difficulty level increases. The difficulty level changes after every 2016th block.

**The mining process**

Note: We will primarily be talking about [Bitcoin mining](http://blockgeeks.com/what-is-bitcoin-mining-an-easy-guide/) here.

When the Bitcoin mining software wants to add a new block to the blockchain, this is the procedure it follows.

* The hash of the contents of the new block is taken.
* A nonce (random string) is appended to the hash.
* The new string is hashed again.
* The final hash is then compared to the difficulty level and seen whether it’s actually less than that or not.
* If not, then the nonce is changed and the process repeats again.
* If yes, then the block is added to the chain and the public ledger is updated and alerted of the addition.
* The miners responsible for this are rewarded with bitcoins.

**What is hash rate?**

Hash rate basically means how fast these hashing operations are taking place while mining. A high hash rate means more people and software machines are taking part in the mining process and as a result, the system is running smoothly. If the hash rate is too fast the difficulty level is increased. If the hash rate becomes too slow then the difficulty level is decreased.

**What is a Public and Private Key Pair?**

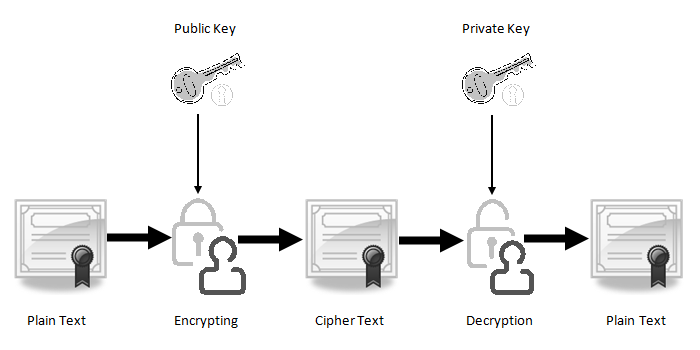
Private Key and public key are a part of [encryption](https://www.ssl2buy.com/wiki/what-is-encryption-and-decryption/) that encodes the information.

Public and Private key pair helps to encrypt information that ensures data is protected during transmission.

Both keys work in two encryption systems called symmetric and asymmetric.

Symmetric encryption (private-key encryption or secret-key encryption) utilize the same key for encryption and decryption.

Asymmetric encryption utilizes a pair of keys like public and private key for better security where a message sender encrypts the message with the public key and the receiver decrypts it with his/her private key.



**Public Key:**

Public key uses asymmetric algorithms that convert messages into an unreadable format. A person who has a public key can encrypt the message intended for a specific receiver. The receiver with the private key can only decode the message, which is encrypted by the public key. The key is available via the public accessible directory.

**Private Key:**

The private key is a secret key that is used to decrypt the message and the party knows it that exchange message.