# What is Blockchain? What are Cryptocurrencies

Sorry for the long break. I have been busy trying to get employed and am glad to say that I have found some work in actuarial analytics (yay?). Although I still plan to continue with the machine learning direction in these paras, there has been a recent buzz regarding Bitcoin and blockchains. I swear if I hear the words 'decentralized' and 'transparency' again...

## Introduction

This para will attempt to explain simply and briefly just what blockchain is and why I think cryptocurrencies are actually a *by-product* of it, instead of a driver. Before we even start to explain what blockchain is, let's first review what life is without it. We'll do this with the help of 3 people:

* Abdul
* Wong
* Chris

The 3 of them are constantly exchanging money amongst themselves and an example of their transactions at any given time may look something like this:

INSERT PICTURE HERE

From the figure, we can see that all the transactions are recorded into a 'common' ledger or database. This means that all 3 people, must believe that the common ledger is official and that there are no errors made by whoever is handling the transactions in this common ledger. There is also an issue if the handler of the common ledger decides to charge a fee for handling and storing all the transactions. This analogy can be extended to the status quo (mostly) in our society where this 'common ledger' can represent financial intermediaries such as banks or credit card companies. This brings us to our current concept of handling transactions: ***Trusting the central system, much like how we trust the banks to correctly display our balances and transactions!*** So, what does blockchain do? In a nutshell, blockchain turns the previous figure into this:

INSERT PICTURE HERE

It allows all 3 of them to hold the transaction records, negating the need for any sort of common ledgers and their respective handlers (or middlemen). If you aren't a full-blown moron, you should have several questions regarding the validity of blockchain such as:

## How do we ensure everyone gets the same ledger?

If a new transaction happens where Abdul pays Chris $20, Abdul can simply broadcast this new updated ledger to everyone in the network. Following this, if Chris now wants to pay $20 to Wong using the money he just got from Abdul, how can he be sure that Wong received the same updated ledger and recognizes that Chris in fact has the $20 to pay him? If we see this from a realistic perspective for just a minute, imagine that you are receiving thousands of different broadcasts from different people regarding their transactions. How can you be sure (like Chris) that everyone else is recording the same transactions in their own personal ledgers in that same order? This brings us to the use of a ***protocol***. A big part of blockchain is a set of steps that everyone follows in order to accept or reject transactions such that if followed, everyone’s' personal ledger will have the same transactions!

## What is this protocol?

The protocol is to ***trust* in the ledger that has the most *computational work* put into it**. Do not be alarmed if you have no clue what that sentence meant. The following sections will explain just that. For now, just accept my claim that if we follow this protocol, any fraudulent activity or conflicting ledgers will require an unrealistic amount of computational work to bring about.

**WARNING: IF YOU INSIST ON CONTINUING TO EXPLAIN BLOCKCHAIN WITH 'DECENTRALISED' AND 'TRANSPARENCY' THEN READ NO FURTHER. THINGS WILL ACTUALLY GET TECHNICAL HERE!**

## Cryptographic Hash Functions

We start to slowly explain the protocol by first defining what exactly *computational work* means. In order to do this, we must define something called a 'Hash function'. As the 'function' part of the name suggests, a hash function takes an input and churns out some sort of output. If you need a refresher on what a function actually is, you can refer to my previous para. Here, the input of a hash function can be any kind of message or file. The output, however, is a **string of bits of a fixed length (101101001...)**, usually converted into a HEX format (a mix of alphabets and digits). An example of this would be:

INSERT PICUTRE HERE

I will not go through the code and algorithms for encryption and hashing, but for a more in depth look into the processes, you can check out the resource [here](https://www.cryptocompare.com/coins/guides/how-does-a-hashing-algorithm-work/). Let's break down what we see in the output above.

* SHA256 is a particular hashing algorithm or function. There exist many different hashing algorithms such as SHA224, SHA384, each with its individual uses and different output lengths.
* The first input that I encrypted is 'Jon' and the second input that I encrypted is 'Jon!' The b in front of the input just converts alphabets into bytes.
* The output is unique (almost all the time) and even the slightest change in input will produce a drastically different output. This can be observed in the vast differences in the 2 outputs even though the only difference between the inputs was an exclamation mark (!)
* Output should be (theoretically) unpredictable as input changes

One thing to note is that these hash functions are a one-way function. This means that it is easy to check the hashed value of 'Jon', but it takes an unrealistic amount of effort to retrieve 'Jon' from the hash value of:

5f39b51ae9a4dacbb8d9538229d726bfb7e1a03633e37d64598c32989a8c1277. Another way to think about this is by mixing colors.

Given red and yellow, we can easily tell that the output will be orange. On the other hand, if we were given orange first, it is extremely hard (or impossible) to tell exactly which mix of colors could have led us to this. This means that the best way to retrieve the initial input from a hashed value is to just guess random combinations of bits until 'Jon' appears. Using SHA256, this amounts to going through 2^256 different combinations. This adds an element of security through encryption. In fact, if you were to go to your internet browser > View > Developer > Developer Tools > Security > View certificate > Details, you will realize that your secured connection is probably encrypted using SHA256!

## What does this have to do with 'Computational Work'?

Let's go back to our original ledger with Abdul, Chris and Wong.

INSERT PICTURE

Much like 'Jon' in the previous example, the transaction data in this ledger can be used as an input into the SHA256 function, which will then output something like:

010101000010100010001000010101010111100101001010...

Recall that any changes, no matter how small, would produce a seemingly random and totally different sequence of output again. What if I wanted my SHA256 output for this ledger to have 10 initial 0 values?

**0000000000**1010100010100100010101001010001001010...

How would I go about doing this and how long do you think it will take me to achieve this? Firstly, recognize from combinatorics that the probability of getting the first 10 values to be zero by a **random draw** is exactly 1/2^10 or 1/1024. Also recall that due to the nature of this encryption, the best way to find this particular hash value is by random guessing or trial and error! So, we implement this additional value called a nonce to our ledger behind our transaction data to test the hashed value until we get our desired 0000000000 as the first 10 digits!

## What is a Nonce?

A nonce is simply an integer that is added to the ledger in order to get a desired hashed value. Recall that any little change would completely change the SHA256 output values, this includes adding a number behind your ledger!

INSERT PICTURE HERE

The figure above depicts the process of increasing the nonce by 1 and checking to see if the hash value has zeros in the first 10 digits. In this particular example, we had to cycle through 821 trials before actually getting our desired hash value! Note that we can easily verify this by inputting the ledger + nonce value of 821 to check if our encrypted hash is indeed 000000000010010010... or mathematically, we check by taking:

**SHA256 (**ledger + 821**)** = **0000000000**101000101100111100...

This means that we can easily check that the required computation is done without having to actually go through all that hard work again, which is a property of the one-way function! This nonce value which leads to the desired hash conditions is also called the ***'Proof of Work' number***.

## So What?

Now that we have an idea of what 'computational work' means, we can go back to defining how this protocol actually works. The first step is to grab a bunch of these floating broadcasted transactions and group them into this thing called a 'block', with its respective proof of work number.

INSERT PICTURE HERE

As of now, a block would contain a number of selected transactions as well as a particular nonce or *proof of work number* which returns a hashed value that has 0 for the first 10 digits. Also, to implement the idea of order and sequence, the hash value of the previous block is also added into the current block's information! This way, if data from any of the transactions is changed, its hash along with all future blocks' hash will also be changed, and the proof of work must be re-computed for all the blocks! Because of the way the blocks are now linked with each other, the term ledger will be replaced with the more fitting and familiar: Blockchain.

INSERT PICTURE HERE

Much like how a contract is not valid without a signature from a person, a block will not be valid or ***official*** and broadcasted to everyone else without a proof of work number. It is important to note that this is the first time we have used the term official in the context of blockchain. This is because making transactions official without having a central system is our goal all along and we now have a way to define what official actually means.

## Who is doing all the computational work?

Technically, anyone and everyone can do it! Let's go back to the scenario where you are just listening to many people broadcasting their transactions.

INSERT PICTURE HERE

What you can do now is to choose a group some of these transactions into a block and do the computational work to get the proof of work number which makes that particular block and its transactions ***official.*** By doing this, you are also 'adding' a new block to the chain and broadcasting it to everyone else. To reward and incentivize these block creators (or more commonly known as *miners*) to keep creating blocks to officialize transactions, a special currency called a **cryptocurrency** is created to remunerate them! The process of creating new blocks is called mining because:

* It requires a lot of work
* It introduces new supply of currency into the market

Much like how real mining works. So, in summary, what a miner really does is:

* Gather and group transactions
* Find the proof of work number
* Make the block (and the transactions inside) official
* Broadcast this update to everyone
* Get paid for their work in cryptocurrencies

Another key step in the protocol is dealing with conflicting block broadcasts. When this happens, the protocol dictates **that the longer chain, or the one with more computational work put into it, will be the official/correct one.**

## What are the special hash values required?

So far, our examples relating to proof of work all require a nonce such that the SHA256 function outputs a string of digits where the first 10 digits are zero, or something like:

**0000000000**1110100010...

What we must understand here is that finding a hash function output with a condition where '*the* ***first*** *value must be 0*' is much easier to achieve compared to a condition where '*the* ***first 30*** *values must be 0*'. In other words, a hash function of:

**0**110010010010011000100101001001011100010100...

Is way easier to obtain through trial and error compared to a hash function like:

**000000000000000000000000000000**1110100010101...

In fact, the probability of getting the first hash output is only 1/2 as the first digit can only be 0 or 1 with any nonce value. On the other hand, the probability of obtaining the second hash output is 1/2^30, or literally one in a billion! This also means that the miner would almost certainly have had to cycle through a billion different nonce values in order to find the right one that gives the 30-zero hash. Once we can understand the different levels of difficulty when changing the conditions, we must now realize the implications of this! If our proof of work condition was easily met (meaning that the number of zeroes required in the hash output is low), miners would be:

* Adding new blocks to the chain very often
* Making new transactions official at a very fast rate
* Getting paid with cryptocurrencies very regularly

Although we used a fixed number of 10 zero digits in our examples, in reality, this number varies and fluctuates in order to regulate the inflow of cryptocurrencies into this virtual economy. Bitcoin for example sets the condition around 30 - 35 zero-digit hashes, which results in a new block created (and BTC created and distributed) approximately once every 10 mins.

**So why bother to regulate hash values?**

Imagine if the hashing conditions were easily met. This could lead to several fundamental and economic problems within the blockchain and cryptocurrency system:

* Cryptocurrencies will be abundant as miners would be able to create new blocks at a very fast rate e.g 1 per second.
* This means that the value of cryptocurrencies would greatly diminish as the supply is now so huge
* When the value of cryptocurrencies go down, miners are no longer incentivized to continue creating blocks as their computational work can be applied elsewhere with better returns
* When miners stop creating blocks, the chain stops and transactions stop becoming official, which was the original purpose of blockchain
* This also goes against the protocol of trusting computational work, as the work required is non-existent

In summary, there must be a balance between **keeping miners excited to continue making transactions official** and **maintaining the value of cryptocurrencies so that miners still find it worth to do so**. This balance is then maintained by tweaking the hashing conditions!

## Why does this protocol work?

Let's try to explain this with an example. Let's say that Wong now tries to create a fraudulent block which contains a transaction that reads:

Abdul pays Wong $2000

This means that Wong, has to start mining for the proof of work number in order to officialize this block faster than the rest of the community who are mining blocks with no fraudulent transactions. If by some luck, Wong manages to get the proof of work number and adds/broadcasts the first block (the one with the fraudulent transaction) before anyone else in the community, he must maintain this position of 'being first' in order to keep his fraudulent transaction because, recall that the protocol dictates everyone to choose the longer than 50% of all the computing power in the world, it is very likely that the community will start to officialize blocks faster than Wong and create a longer chain, thus negating his fraudulent transaction. One implication of this is that, we should not trust the transactions within a block that has just been added. In fact, in order to be certain, we should actually wait for a few more blocks to be added to be sure that we are actually looking at the community's chain instead of Wong's fraudulent chain! With this condition, as well as using previous block's hash in the current block's information, we have managed to establish a protocol in which everyone will get the same ordered personal ledger in the community without have a central system! For a more in depth explanation of the underlying statistics and codes, you can refer to the original Bitcoin white paper [here](https://bitcoin.org/bitcoin.pdf).

## So, what is blockchain?

Recall the concept of trusting the central system. The handler is the sole body that makes transactions official. With blockchain, this job of making transactions official is now delegated to many different people in the form of a race to create new blocks and is incentivized by the payment of cryptocurrencies! This is the underlying reason for its decentralization and transparency! Here are some summarized differences between the concept of centralized and decentralized systems:

|  |  |  |
| --- | --- | --- |
|  | **Centralized System** | **Blockchain** |
| **Regulations** | Statutory laws | Protocols |
| **Definition of official** | As long as it is in the central ledger | When transaction in a block have a proof of work number |
| **Placement of trust** | In the system and governing authorities | In computational work and code |
| **Location of the ledger** | In a single database | In everyone’s personal records as long as they are in the network |
| **Remuneration for the ledger handlers** | Fees and charges set by the handler in nominal currencies | Paid in cryptocurrencies for the people who help build the chain |

Note that this entry is by no means a comprehensive guide to blockchain and cryptocurrencies. There are underlying concepts and algorithms such as Merkle trees, consensus algorithms and different versions of 'proof of work' that isn't covered due to the level of technicality involved, and also does not cover the viability of cryptocurrencies as a financial instrument due to the fact that I don't give a rat's ass. Hopefully next time someone asks you about blockchain, you will either say 'I don't know' or give a better explanation than 'Decentralized', 'Transparent' and 'Encryption'.