ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Φυσικής

DIPLOMA THESIS

Blockchain Technology

Author : Kappos Angelos Dionysios

Supervisor : Stasinopoulos Georgios , Professor , National Technical University of Athens - Division of Communication, Electronic and Information Engineering

Athens 2019

Kappos A. (2019) Blockchain Technology . National and Kapodistrian University of Athens, Faculty of Physics , Department of Electronics , Computers, Telecommunications and Control, Master Degree Program in Control and Computing , Master’s Thesis .

ABSTRACT

Blockchain is a new open-source technology which has a huge potential for implementation in a variety of industries, which was initially released as the underlying technology for the world’s first decentralized global digital currency, Bitcoin.

Blockchain is a constantly growing and secure ledger that keeps a permanent record of all the transactions that have taken place, in a secure chronological and immutable way. Furthermore, that information is preserved using highly advanced cryptography. All the information that's captured in that blockchain is captured in a chronological manner.

AIMS – THESIS PURPOSE

The main goal of this master thesis' research:

- is to investigate how the blockchain works
- its main benefits and advantages
- to clarify the weaknesses of this ledger
- define the concept of Blockchain
- analyze use cases and implications of the technology.

Additionally, the study provides information about general implications of Blockchain and the abilities for developing these areas, and a technical implementation of all the included chapters in this thesis' research in order to show practically the way to develop a blockchain .

To my father,
Dimitrios K. Kappos,
Who showed me which is the meaning of “brightness” in our lifes,
How to fight adversity and to persist no matter what.

Πίνακας περιεχομένων

CHAPTER 1 Introduction

What is NOT Blockchain?

Many of us have heard many definitions of what Blockchain is. The answer to this simple question varies. Firstly, a possible way to understand this definition is to clarify what blockchain is not:

- An application
- A technology
- An algorithm
- A method of encryption
- A proprietary product.

It utilizes some of the above, but it doesn't come – in and of itself – in a vendor stamped package. So, with this information, it's essential to address what blockchain is and how it works.

What is it, Blockchain?

A blockchain is a constantly growing and secure ledger that keeps a permanent record of all the transactions that have taken place, in a secure chronological and immutable way. There's not just one blockchain, it's actually distributed and there are thousands of copies all around the world for it. Furthermore, that information is preserved using highly advanced cryptography. All the information that's captured in that blockchain is captured in a chronological manner.

A blockchain is, in the simplest of terms, a time-stamped series of immutable record of data that is managed by cluster of computers not owned by any single entity. Each of these blocks of data (i.e. block) are secured and bound to each other using cryptographic principles (i.e. chain).

That means that every block that gets added to the blockchain is added as a new block , and chronologically ordered. As you will see the numbers of the blocks are growing as new blocks are added. Finally, all the information that's recorded on the blockchain is immutable. That means that nobody can change it once it's actually recorded inside.

It is indisputably a rare change to information storage and processing plumbing. In essence it hardwires the history of information. Be that information with it’s own inherent value (e.g. cryptocurrency and intellectual property), or information about asset ownership, and ownership changes.

Blockchains are also, in their basic form, open source. A piecing together of earlier open source concepts and code as outlined in a now famous paper written by Bitcoin’s founder, Satoshi Nakamoto [9].

How does it do all that?

Transaction data is written into a block (in computing terms, that’s a small chunk of space on a disc in a computer somewhere, just like all the other data in the world). *What **distinguishes** the block from everything else you create and save are the tamper resistant timestamps and unique identifiers*: A transaction includes a hashed record of recent valid transactions, summary transaction info, address of the creator (also usually cryptographically hashed), block timestamp, and hash of the block preceding it (hence the chain). **Hashes**, very simply are alphanumeric strings generated by applying an algorithm to block contents.

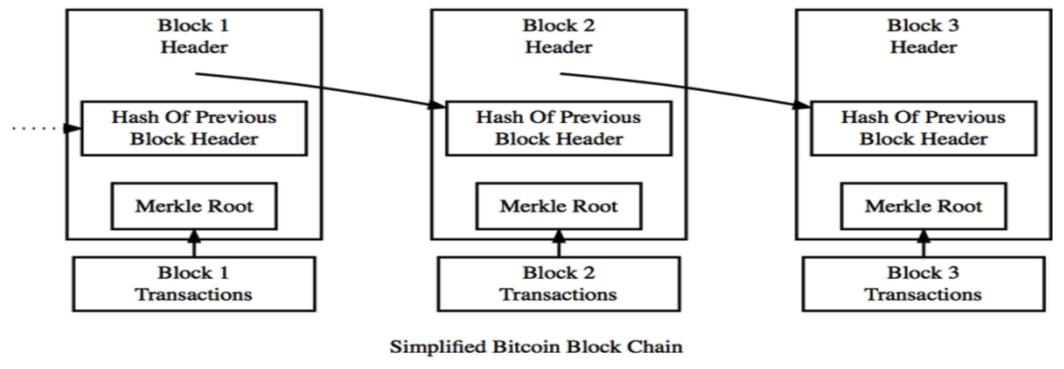
A new **block** can only be created when certain criteria are met. Criteria based around the algorithm. **Block miners** run calculations using big numbers for as long as it takes to produce a hash that meets chain creator requirements. That takes significant **processing power**. Criteria for cryptocurrency have also been designed to future proof that difficulty. If hashes are found too fast (Bitcoins are produced at a rate of roughly 1 every 10 minutes), the requirements get harder.

Hashes can be, with sufficient processing power and time, *reverse engineered* without private keys to get at the hidden plaintext. But, as things currently stand (assuming robust chain creation criteria, evolution of criteria to match future processing capability, and proper implementation), you'd likely die before you succeed, that's why miners are getting important rewards for their job.

Once written and added to the chain it would therefore take significant money and time to replace blocks. And (to correct errors, or add information), you have to replace them. Blocks don't change. The fact of the replacement is recorded in the chain along with the new information.

Additionally, many more important information will be written in the following chapters in depth for every part which consist the Blockchain.

Picture 1. A representation of a simple Blockchain of three blocks.



Is Blockchain Technology the New Internet?

The blockchain is an undeniably ingenious invention – the brainchild of a person or group of people known by the pseudonym, Satoshi Nakamoto. But since then, it has evolved into something greater, and the main question every single person is asking is: What is Blockchain?

*By allowing digital information to be distributed but **not copied***, blockchain technology created the backbone of a new type of internet. Originally devised for the digital currency, Bitcoin, the tech community has now found other potential uses for the technology.

So, what is so special about it and why are we saying that it has industry disrupting capabilities?

The blockchain network has **no central authority** — it is the very definition of a democratized system. Since it is a shared and immutable ledger, the information in it is open for anyone and everyone to see. Hence, anything that is built on the blockchain is by its very nature transparent and everyone involved is accountable for their actions.

Blockchain Explained

A blockchain carries *no transaction cost*. (An infrastructure cost yes, but no transaction cost.) The blockchain is a simple yet ingenious way of passing information **from A to B** in a fully automated and safe manner. One party to a transaction initiates the process by creating a block. This block is verified by thousands, perhaps millions of computers distributed around the net. The verified block is added to a chain, which is stored across the net, creating not just a unique record, but **a unique record with a unique history**. *Falsifying a single record would mean falsifying the entire chain in millions of instances*. That is virtually impossible. Bitcoin uses this model for monetary transactions, but it can be deployed in many others ways.

Think of a railway company. We buy tickets on an app or the web. The credit card company takes a cut for processing the transaction. With blockchain, not only can the railway operator save on credit card processing fees, it can move the entire ticketing process to the blockchain. The two parties in the transaction are the railway company and the passenger. The ticket is a block, which will be added to a ticket blockchain. Just as a monetary transaction on blockchain is a unique, independently verifiable and unfalsifiable record (like Bitcoin), so can your ticket be. Incidentally, the final ticket blockchain is also a record of all transactions for, say, a certain train route, or even the entire train network, comprising every ticket ever sold, every journey ever taken.

But the key here is this: **it's free**. Not only can the blockchain transfer and store money, but it can also replace all processes and business models which rely on charging a small fee for a transaction. Or any other transaction between two parties.

Perhaps most profoundly, blockchain promises to democratize & expand the global financial system. Giving people who have limited exposure to the global economy, **better**

access to financial and payment systems and stronger protection against corruption and exploitation.

The potential impacts of blockchain technology on society and the global economy are hugely significant. With an ever-growing list of real-world uses, blockchain technology promises to have a massive impact.

In the financial world the applications are more obvious and the revolutionary changes more imminent. Blockchains will change the way stock exchanges work, loans are bundled, and insurances contracted. They will eliminate bank accounts and practically all services offered by banks. Almost every financial institution will go bankrupt or be forced to change fundamentally, once the advantages of a safe ledger without transaction fees is widely understood and implemented. After all, the financial system is built on taking a small cut of your money for the privilege of facilitating a transaction. Bankers will become mere advisers, not gatekeepers of money. Stockbrokers will no longer be able to earn commissions and the buy/sell spread will disappear. This is just the beginning and only one small part of the impact that Blockchain will do in the global Financial Systems, because it will affect other domains in our common life that have middle role in our transactions, which will be described in detail in our Chapter Blockchain Use Cases.

How Does Blockchain Work?

Picture a spreadsheet that is duplicated thousands of times across a network of computers. Then imagine that this network is designed to regularly update this spreadsheet and you have a basic understanding of the blockchain.

Information held on a blockchain exists as a shared — and continually reconciled — database. This is a way of using the network that has obvious benefits. <u>The blockchain database isn’t stored in any single location</u>, meaning the records it keeps are truly public and easily verifiable<u>. No centralized version</u> of this information exists for a hacker to corrupt. Hosted by millions of computers simultaneously, its data is accessible to anyone on the internet.

The reason why the blockchain has gained so much admiration is that:

- It is not owned by a single entity, hence it is ***decentralized.***

- The data is cryptographically stored inside.

- The blockchain is ***immutable***, so no one can tamper with the data that is inside the blockchain.

- The blockchain is ***transparent*** so one can track the data if they want to.

What's under the blockchain hood?

Simply put, a blockchain is a type of distributed ledger or decentralized database that keeps continuously updated digital records of who owns what. Rather than having a central administrator like a traditional database, (think banks, governments & accountants), a distributed ledger has a network of replicated databases, synchronized via the internet and visible to anyone within the network. Blockchain networks can be private with restricted membership similar to an intranet, or public, like the Internet, accessible to any person in the world.

When a digital transaction is carried out, it is grouped together in a cryptographically protected block with other transactions that have occurred in the last 10 minutes and sent out to the entire network. Miners (members in the network with high levels of computing power) then compete to validate the transactions by solving complex coded problems. [1] The first miner to solve the problems and validate the block receives a reward.

The validated block of transactions is then timestamped and added to a chain in a linear, chronological order. New blocks of validated transactions are linked to older blocks, making a chain of blocks that show every transaction made in the history of that blockchain [2]. The entire chain is continually updated so that every ledger in the network is the same, giving each member the ability to prove who owns what at any given time.

Blockchain's decentralized, open & cryptographic nature **allow** people to **trust each other and transact peer to peer**, *making the need for intermediaries obsolete*. This also brings unprecedented security benefits. Hacking attacks that commonly impact large centralized intermediaries like banks would be virtually impossible to pull off on the blockchain. For example—if someone wanted to hack into a particular block in a blockchain, a hacker would not only need to hack into that specific block, but all of the proceeding blocks going back the entire history of that blockchain. And they would need to do it on every ledger in the network, which could be millions, simultaneously.[12]

The Three Pillars of Blockchain Technology

The three main properties of the Blockchain Technology which has helped it gain widespread acclaim are as follows:

• Decentralization

• Transparency

• Immutability

<u>Pillar #1: Decentralization</u>

You have a centralized entity which stored all the data and you'd have to interact solely with this entity to get whatever information you required.

Another example of a centralized system is banks. They store all your money, and the only way that you can pay someone is by going through the bank.

Now, centralized systems have treated us well for many years, however, they have several vulnerabilities.

• Firstly, because they are centralized, all the data is stored in one spot. This makes them easy target spots for potential hackers.

• If the centralized system were to go through a software upgrade, it would halt the entire system

• What if the centralized entity somehow shut down for whatever reason? That way nobody will be able to access the information that it possesses

• Worst case scenario, what if this entity gets corrupted and malicious? If that happens then all the data that is inside the blockchain will be compromised.

<u>Pillar #2: Transparency</u>

One of the most interesting and misunderstood concepts in the blockchain technology is "transparency." Some people say that blockchain gives you privacy while some say that it is transparent. Why do you think that happens?

Notably, a person's identity is hidden via complex cryptography and represented only by their public address.

<u>Pillar #3: Immutability</u>

Immutability, in the context of the blockchain, means that once something has been entered into the blockchain, it cannot be tampered with

The reason why the blockchain gets this property is that of cryptographic hash function.

Briefly, 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) which gives an output of a fixed length.

In the case of SHA-256, 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.

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.

The blockchain is a linked list which contains data and a hash pointer which points to its previous block, hence creating the chain. What is a hash pointer? 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.

This one small tweak is what makes blockchains so amazingly reliable and trailblazing. 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.

Maintaining the Blockchain – Network and Nodes

The blockchain is maintained by a peer-to-peer network. The network is a collection of nodes which are interconnected to one another. Nodes are individual computers which take in input and performs a function on them and gives an output. The blockchain uses a special kind of network called "peer-to-peer network" which partitions its entire workload between participants, who are all equally privileged, called "peers". There is no longer one central server, now there are several distributed and decentralized peers.

Why do people use the peer-to-peer network?

One of the main uses of the peer-to-peer network is file sharing, also called torrenting. If you are to use a client-server model for downloading, then it is usually extremely slow and entirely dependent on the health of the server. Plus, as mentioned above, it is prone to censorship.

However, in a peer-to-peer system, there is no central authority, and hence if even one of the peers in the network goes out of the race, you still have more peers to download from. Additionaly, it is not subject to the idealistic standards of a central system, hence it is not prone to censorship.

The use of networks and nodes in cryptocurrencies.

The peer-to-peer network structure in cryptocurrencies is structured according to the consensus mechanism that they are utilizing. For cryptos like Bitcoin and Ethereum which uses a normal proof-of-work consensus mechanism (Ethereum will eventually move on to Proof of Stake), all the nodes have the same privilege. The idea is to create an egalitarian network. The nodes are not given any special privileges; however, their functions and degree of participation may differ. There is no centralized server/entity, nor is there any hierarchy. It is a flat topology.

These decentralized cryptocurrencies are structured like that is because of a simple reason, to stay true to their philosophy. The idea is to have a currency system, where everyone is treated as an equal and there is no governing body, which can determine the value of the currency based on a whim. This is true for both bitcoin and Ethereum.

Now, if there is no central system, how would everyone in the system get to know that a certain transaction has happened? The network follows the gossip protocol. Think of how gossip spreads. Suppose Alice sent 3 ETH to Bob. The nodes nearest to her will get to know of this, and then they will tell the nodes closest to them, and then they will tell their neighbors, and this will keep on spreading out until everyone knows. Nodes are basically your nosy, annoying relatives.

So, what is a node in the context of Ethereum? A node is simply a computer that participates in the Ethereum network. This participation can be in three ways

- By keeping a shallow-copy of the blockchain aka a Light Client
- By keeping a full-copy of the blockchain aka a Full Node
- By verifying the transactions aka Mining

However, the problem with this design is that it is not really that scalable. Which is why, a lot of new generation cryptocurrencies adopt a leader-based consensus mechanism. In EOS, Cardano, Neo etc. the nodes elect leader nodes or “super nodes” who are in charge of the consensus and overall network health. These cryptos are a lot faster but they are not the most decentralized of systems.

So, in a way, cryptos have to make the trade-off between speed and decentralization.

How Does the Blockchain Work?

A great example of how Blockchain works:

Imagine you and I bet $50 on tomorrow's weather in San Francisco. I bet it will be sunny, you that it will rain. Today we have three options to manage this transaction:

1. We can trust each other. Rainy or sunny, the loser will give $50 to the winner. If we are friends, this could be a good way of managing it. However, friends or strangers, one can easily not pay the other.

2. We can turn the bet into a contract. With a contract in place both parties will be more prone to pay. However, should either of the two decide not to pay, the winner will have to pay additional money to cover legal expenses and the court case might take a long time. Especially for a small amount of cash, this doesn't seem like the optimal way to manage the transaction.

3. We can involve a neutral third party. Each of us gives $50 to a third party, who will give the total amount to the winner. But hey, she could also run away with all our money. So we end up with one of the first two options: trust or contract.

Neither trust nor contract is an optimal solution: We can't trust strangers, and enforcing a contract requires time and money. The blockchain technology is interesting because it offers us a third option which is secure, quick, and cheap.

Blockchain allows us to write a few lines of code, a program running on the blockchain, to which both of us send $50. This program will keep the $100 safe and check tomorrow's weather automatically on several data sources. Sunny or rainy, it will automatically transfer the whole amount to the winner. Each party can check the contract logic, and once it's running on the blockchain it can't be changed or stopped. This may be too much effort for a $50 bet, but imagine selling a house or a company.

This piece explains how the blockchain works without discussing the technical details in depth, but by digging just enough to give you a general idea of the underlying logic and mechanisms.

The most known and discussed application of the blockchain technology is bitcoin, a digital currency that can be used to exchange products and services.

Bitcoin gives us, for the first time, a way for one Internet user to transfer a unique piece of digital property to another Internet user, such that the transfer is guaranteed to be safe and secure, everyone knows that the transfer has taken place, and nobody can challenge the legitimacy of the transfer. The consequences of this breakthrough are hard to overstate. To keep track of the amount of bitcoin each of us owns, the blockchain uses a ledger, a digital file that tracks all bitcoin transactions. The ledger file is not stored in a central entity server, like a bank, or in a single data center. It is distributed across the world via a network of private computers that are both storing data and executing computations. Each of these computers represents a "node" of the blockchain network and has a copy of the ledger file.

If David wants to send bitcoins to Sandra, he broadcasts a message to the network that says the amount of bitcoin in his account should go down by 5 BTC, and the amount in Sandra's account should increase by the same quantity. Each node in the network will receive the message and apply the requested transaction to its copy of the ledger, updating the account balances.

The fact that the ledger is maintained by a group of connected computers rather than by a centralized entity like a bank has several implications:

- In our bank system we only know our own transactions and account balances; on the blockchain everyone can see everyone else's transactions.

- While you can generally trust your bank, the bitcoin network is distributed and if something goes wrong there is no help desk to call or anyone to sue.

- The blockchain system is designed in such a way that no trust is needed; security and reliability are obtained via special mathematical functions and code.

To send bitcoin you need to prove that you own the private key of a specific wallet as you need the key to encrypt your transaction request message. Since you broadcast the message only after it has been encrypted, you never have to reveal your private key.

Tracking Your Wallet Balance

Each node in the blockchain is keeping a copy of the ledger. So, how does a node know your account balance? The blockchain system doesn't keep track of account balances at all; it only records each and every transaction that is verified and approved. The ledger in fact does not keep track of balances, it only keeps track of every transaction broadcasted within the bitcoin This "balance" verification is performed based on links to previous transactions. In order to send 10 bitcoins to John, Mary has to generate a transaction request that includes links to previous incoming transactions that add up to at least 10 bitcoins. So, how can the system trust that input transactions are valid? It checks all the previous transactions correlated to the wallet you use to send bitcoins via the input references. To speed up the verification process, a special record of unspent transactions is kept by the network nodes. Thanks to this security check, it is not possible to double-spend bitcoins.

Owning bitcoins means that there are transactions written in the ledger that point to your wallet address and haven't been used as inputs yet. All the code to perform transactions on the bitcoin network is open source; this means that anyone with a laptop and an internet connection can operate transactions. However, should there be a mistake in the code used to broadcast a transaction request message, the associated bitcoins will be permanently lost.

Remember that since the network is distributed, there is no customer support to call nor anyone who could help you restore a lost transaction or forgotten wallet password. For this reason, if you are interested in transacting on the bitcoin network, it's a good idea to use the open source and official version of bitcoin wallet software (such as Bitcoin Core), and to store your wallet's password or private key in a very safe repository.

But Is It Really Safe? And Why Is It Called Blockchain?

The bitcoin network allows you to generate as many wallets as you like, each with its own private and public keys. This allows you to receive payments on different wallets, and there is no way for anyone to know that you own all these wallets' private keys, unless you send all the received bitcoins to a single wallet.

The total number of possible bitcoin addresses is 2^{160} or 1461501637330902918203684832716283019655932542976. This large number protects the network from possible attacks while allowing anyone to own a wallet.

Transactions are passed from node to node within the network, so the order in which two transactions reach each node can be different. An attacker could send a transaction, wait for the counterpart to ship a product, and then send a reverse transaction back to his own account. In this case, some nodes could receive the second transaction before the first and therefore consider the initial payment transaction invalid, as the transaction inputs would be marked as already spent. How do you know which transaction has been requested first? It's not secure to order the transactions by timestamp because it could easily be counterfeit. Therefore, there is no way to tell if a transaction happened before another, and this opens up the potential for fraud. If this happens, there will be disagreement among the network nodes regarding the order of transactions each of them received. So the blockchain system has been designed to use node agreement to order transactions and prevent the fraud described above.

The bitcoin network orders transactions by grouping them into blocks; each block contains a definite number of transactions and a link to the previous block. This is what puts one block after the other in time. Blocks are therefore organized into a time-related chain . that gives the name to the whole system: blockchain.

Since any node can suggest a new block, how does the system agree on which block should be the next?

In order to be added to the blockchain, each block must contain the answer to a complex mathematical problem created using an irreversible cryptographic hash function. The only way to solve such a mathematical problem is to guess random numbers that, combined with the previous block content, generate a defined result. It could take about a year for a typical computer to guess the right number and solve the mathematical problem. However, due to the large number of computers in the network that are guessing numbers, a block is solved on average every 10 minutes. The node that solves the mathematical problem acquires the right to place the next block on the chain and broadcast it to the network.

And what if two nodes solve the problem at the same time and send their blocks to the network simultaneously? In this case, both blocks are broadcast and each node builds on the block that it received first. However, the blockchain system requires each node to build immediately on the longest blockchain available. So if there is ambiguity about which is the last block, as soon as the next block is solved, each node will adopt the longest chain as the only option.

Due to the low probability of solving blocks simultaneously, it's almost impossible that multiple blocks would be solved at the same time over and over, building different "tails," so the whole blockchain stabilizes quickly to one single string of blocks that every node agrees on.

A disagreement about which block represents the end of the chain tail opens up the potential for fraud again. If a transaction happens to be in a block that belongs to a shorter tail once the next block is solved, this transaction, along with all others in its block, will go back to the unconfirmed transactions.

***Using blockchain technology has remarkable benefits*:**

• You have complete control of the value you own; there is no third party that holds your value or can limit your access to it.

• The cost to perform a value transaction from and to anywhere on the planet is very low. This allows micropayments.

• Value can be transferred in a few minutes, and the transaction can be considered secure after a few hours, rather than days or weeks.

• Anyone at any time can verify every transaction made on the blockchain, resulting in full transparency.

• It's possible to leverage the blockchain technology to build decentralized applications that would be able to manage information and transfer value fast and securely.

However, there are a few challenges that need to be addressed:

• Transactions can be sent and received anonymously. This preserves user privacy, but it also allows illegal activity on the network.

• Though many exchange platforms are emerging, and digital currencies are gaining popularity, it's still not easy to trade bitcoins for goods and services.

• Bitcoin, like many other cryptocurrencies, is very volatile: There aren't many bitcoins available in the market and the demand is changing rapidly. Bitcoin price is erratic, changing based on large events or announcements in the cryptocurrencies industry.

Overall, the blockchain technology has the potential to revolutionize several industries, from advertising to energy distribution. Its main power lies in its decentralized nature and ability to eliminate the need for trust.

New use cases are arising all the time—like the possibility of creating a fully decentralized platform that runs smart contracts like Ethereum. But it's important to remember that the technology is still in its infancy. New tools are being developed every day to improve blockchain security while offering a broader range of features, tools, and services.

CONSENSUS: How do you resolve conflicts?

A common conflict is when multiple miners create blocks at roughly the same time. Because blocks take time to be shared across the network, which one should count as the legit block?

Example.

Let's say all the nodes on the network have synchronized their blockchains, and they are all on block number 80.

If three miners across the world create 'Block 81' at roughly the same time, which 'Block 81' should be considered valid? Remember that each 'Block 81' will look slightly different: They will certainly contain a different payment address for the 25 BTC block reward; and they may contain a different set of transactions. Let's call them 81a, 81b, 81c.

Which block should count as the legit one?

How do you resolve this?

Longest chain rule. In bitcoin, the conflict is resolved by a rule called the "longest chain rule".

In the example above, you would assume that the first ‘Block 81’ you see is valid. Let’s say you see 81a first. You can start building the next block on that, trying to create 82a:

Treat the first block you see as legitimate.

However, in a few seconds you may see 81b. If you see this, you keep an eye on it. If later you see 82b, the “longest chain rule” says that you should regard the longer ‘b’ chain as the valid one (…80, 81b, 82b) and ignore the shorter chain (…80, 81a). So, you stop trying to make 82a and instead start trying to make 83b:

Longest chain rule: If you see multiple blocks, treat the longest chain as legitimate.

The “longest chain rule” is the rule that the bitcoin blockchain ecosystem uses to resolve these conflicts which are common in distributed networks.

DEFENCE: How do you make it hard for baddies?

A problem with a permissionless, or open networks is that they can be attacked by anyone. So there needs to be a way of making the network-as-a-whole trustworthy, even if specific actors aren’t.

What can and can’t miscreants do?

A dishonest miner can:

1. Refuse to relay valid transactions to other nodes

2. Attempt to create blocks that include or exclude specific transactions of his choosing

3. Attempt to create a ‘longer chain’ of blocks that make previously accepted blocks become ‘orphans’ and not part of the main chain

He can’t:

1. Create bitcoins out of thin air*

2. Steal bitcoins from your account

3. Make payments on your behalf or pretend to be you

That’s a relief.

*Well, he can, but only his version of the ledger will have these transactions. Other nodes will reject this, which is why it is important to confirm a transaction across a number of nodes.

With transactions, the effect a dishonest miner can have is very limited. If the rest of the network is honest, they will reject any invalid transactions coming from him, and they will hear about valid transactions from other honest nodes, even if he is refusing to pass them on.

With blocks, if the miscreant has sufficient block creation power (and this is what it all hinges on), he can delay your transaction by refusing to include it in his blocks. However, your transaction will still be known by other honest nodes as an ‘unconfirmed transaction’, and they will include it in their blocks.

Worse though, is if the miscreant can create a longer chain of blocks than the rest of the network and invoking the “longest chain rule” to kick out the shorter chains. This lets him unwind a transaction.

Here’s how you can do it:

1. Create two payments with the same bitcoins: one to an online retailer, the other to yourself (another address you control)

2. Only broadcast the payment that pays the retailer

3. When the payment gets added in an honest block, the retailer sends you goods

4. Secretly create a longer chain of blocks which excludes the payment to the retailer, and includes the payment to yourself

5. Publish the longer chain. If the other nodes are playing by the “longest chain rule” rule, then they will ignore the honest block with the retailer payment and continue to build on your longer chain. The honest block is said to be ‘orphaned’ and does not exist to all intents and purposes.

6. The original payment to the retailer will be deemed invalid by the honest nodes because those bitcoins have already been spent (in your longer chain).

CHAPTER 2 SHA256 - Features of Hashing Algorithms

Purpose of Cryptographic Hash Algorithms

A hash function is a function which takes an arbitrary length input and produces a fixed length 'fingerprint' string. Common usage of such a function is as index into a hashtable. Cryptographic hash functions have several additional properties which makes them suitable to use as a means to check the integrity of a message and as part of digital signature schemes. Such a scheme consists of a secret key ks, a public key kp and two functions Sign (M, ks), which produces signature S, and Verify(M, S, kp), which returns a boolean indicating if the given S is a valid signature for message M. Such a signature should prove the authenticity and integrity of the message; Such that we can be sure it really was the sender of the message who sent it, and that it really was this message he sent. Finally, we can use the signature to prove that the sender sent it and no one else could have done so: non-repudiation

A function requirement is that Verify(M,Sign(M,ks),kp) = true for a key pair (ks,kp). On the other hand, it should be impossible to create a forged signature.

Short History [2]

The Secure Hash Algorithm (SHA) was developed by the NIST in association with the NSA and first published in May 1993 as the Secure Hash Standard. The first revision to this algorithm was published in 1995 due to a unpublished flaw found, and was called SHA-1. The first version, later dubbed SHA-0, was withdrawn by the NSA. The SHA hash function is similar to the MD4 hash function, but adds some complexity to the algorithm and the block size used was changed. SHA was originally intended as part of the Digital Signature Standard (DSS), a scheme used for signing data and needed a hash function to do so.

In addition to the SHA-1 hash, the NIST also published a set of more complex hash functions for which the output ranges from 224 bit to 512 bit. These hash algorithms, called SHA-224, SHA-256, SHA-384 and SHA-512 (sometimes referred to as SHA-2) are more complex because of the added non-linear functions to the compression function. As of

January 2008, there are no attacks known better than a brute force attack. Fortunately, the SHA-2 hash functions produce longer hashes, making a feasible attack more difficult. Consider for example the SHA-512 hash function, producing 512 bit hashes and thus having an approximate complexity against collisional attacks of 2256. Even if the logarithmic complexity would be halved (from 256 to 128), this would still be out of reach for practical purposes for the coming decade or so.

Understanding a SHA256 Hash

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) which gives an output of a fixed length.

As you can see, in the case of SHA-256, 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. Before we go any further we need to first see the various properties of hashing functions and how they get implemented in the blockchain.

Cryptographic hash functions

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. Let’s run through them one by one.

Property 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. This is critical because if you get different hashes every single time it will be impossible to keep track of the input.

Property 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.

Property 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. Notice the use of the word "infeasible" instead of "impossible". We already know that it is not impossible to determine the original input from its hash value. Let's take an example.

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". 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, what will happen if you use this method?

- **Best case scenario**: You get your answer on the first try itself. You will seriously have to be the luckiest person in the world for this to happen. The odds of this happening are astronomical.

- **Worst case scenario**: You get your answer after 2^128 – 1 times. Basically, it means that you will find your answer at the end of all the data.

- **Average scenario**: You will find it somewhere in the middle so basically after 2^128/2 = 2^127 times. To put that into perspective, 2^127 = 1.7 X 10^38. In other words, it is a huge number.

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

Property 4: Small Changes In The Input Changes the Hash.

Even if you make a small change in your input, the changes that will be reflected in the hash will be huge. This is a critical function because this property of hashing leads to one of the greatest qualities of the blockchain, its immutability.

Property 5: 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). What that means is that for the most part, each input will have its own unique hash. Why did we say"for the most part"? Let's talk about an interesting concept called "The Birthday Paradox". What is the Birthday Paradox?

If you meet any random stranger out on the streets the chances are very low for both of you to have the same birthday. In fact, assuming that all days of the year have the same likelihood of having a birthday, the chances of another person sharing your birthday is 1/365 which is a 0.27%. In other words, it is really low.

However, having said that, if you gather up 20-30 people in one room, the odds of two people sharing the exact same birthday rises up astronomically. In fact, there is a 50-50 chance for 2 people of sharing the same birthday in this scenario!

Why does that happen? It is because of a simple rule in probability which goes as follows. Suppose you have N different possibilities of an even happening, then you need square root of N random items for them to have a 50% chance of a collision.

So applying this theory for birthdays, you have 365 different possibilities of birthdays, so you just need Sqrt (365), which is ~23~, randomly chosen people for 50% chance of two people sharing birthdays.

What is the application of this in hashing?

Suppose you have a 128-bit hash which has 2^128 different possibilities. By using the birthday paradox, you have a 50% chance to break the collision resistance at the sqrt (2^128) = 2^64th instance.

As you can see, it is much easier to break collision resistance than it is to break preimage resistance. No hash function is collision free, but it usually takes so long to find a collision. So, if you are using a function like SHA-256, it is safe to assume that if H(A) = H(B) then A = B.

Property 6: Puzzle Friendly

Now, this is a fascinating property, and the application and impact that this one property has had on cryptocurrency are huge (more on that later when we cover mining and crypto puzzles). First let's define the property, after that we will go over each term in detail.

For every output "Y", if k is chosen from a distribution with high min-entropy it is infeasible to find an input x such that H(k|x) = Y.

That probably went all over your head! But it's ok, let's now understand what that definition means.

What is the meaning of "high min-entropy"?

It means that the distribution from which the value is chosen is hugely distributed so much so that us choosing a random value has negligible probability. Basically, if you were told to choose a number between 1 and 5, that's a low min-entropy distribution. However, if you were to choose a number between 1 and a gazillion, that is a high min-entropy distribution

What does "k|x" mean?

The "|" denotes concatenation. Concatenation means adding two strings together. Eg. If I were to concatenate "BLUE" and "SKY" together, then the result will be "BLUESKY".

So now let's revisit the definition.

Suppose you have an output value “Y”. If you choose a random value “k” from a wide distribution, it is infeasible to find a value X such that the hash of the concatenation of k and x will give the output Y.

Once again, notice the word “infeasible”, it is not impossible because people do this all the time. In fact, the whole process of mining works upon this (more on that later).

Hashing and data structures

A data structure is a specialized way of storing data. There are two data structure properties that are critical if you want to understand how a blockchain works. They are:

1. **Pointers.**

2. **Linked Lists.**

It is a sequence of blocks, each containing data which is linked to the next block via a pointer. The pointer variable, in this case, contains the address of the next node in it and hence the connection is made. The last node, as you can see, has a null pointer which means that it has no value.

One important thing to note here, the pointer inside each block contains the address of the next block. That is how the pointing is achieved. Now you might be asking what does that mean for the first block in the list? Where does the pointer of the first block stay?

The **first block** is called the “**genesis block**” and its pointer lies out in the system itself.

If you are wondering what the “hash pointer” means, we will get there in a bit.

As you may have guessed by now, this is what the structure of the blockchain is based on. A block chain is basically a linked list.

The blockchain is a linked list which contains data and a hash pointer which points to its previous block, hence creating the chain. What is a hash pointer? 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. This one small tweak is what makes blockchains so amazingly reliable and trailblazing.

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.

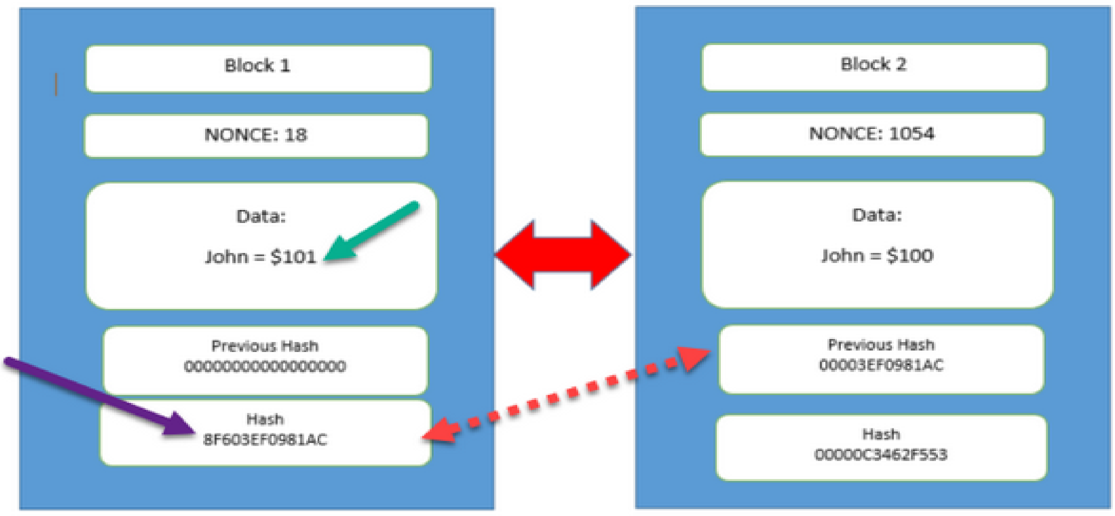
<u>A block header contains:</u>

- 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.

Right now, let’s focus on the Hash of the Merkle Root. But before that, we need to understand *what a Merkle Tree is [Chapter 8].*

CHAPTER 3 Immutable Ledger

An Immutable Ledger simply means a record that cannot be changed. The idea behind all of this is data security and proof that the data has not been altered. Why are we so concerned here? In a blockchain application like BitCoin, we are tracking transactions of money. Imagine if you sent me an electronic funds transfer for $100. How would you feel if I hacked into your bank and changed that $100, to $100,000? you want to make sure that when you set up a transfer for $100, no one can alter it. With blockchain, that is true even if you want it to be altered (you made a mistake). If you want to fix an error, you will have to add another transaction to the blockchain to correct the issue. This is actually good accounting practice, as once an entry is made into a ledger, it should never be removed or altered. So how does blockchain ensure immutability of the ledger? It all resides in the concept of the hash. If hacker tries to alter anything in the block below, its hash will change. Now the hash will no long match the previous hash in the second block. So, the hacker would have to change the next block, and the block after that, etc. And even if they were able to pull that off, remember that the blockchain resides on multiple computers. The hacker would need to make all of these changes simultaneously. When you consider the millions of nodes that make up a blockchain environment like BitCoin, you will see that would be impossible.



Immutability has the potential to transform the auditing process into a quick, efficient, and cost-effective procedure, and bring more trust and integrity to the data businesses use and share every day. Blockchain implementation can bring an unprecedented level of trust to the data enterprises use on a daily basis—immutability provides integrity (both in its technical and primary definition). With blockchain, we can prove to our stakeholders that the information we present and use has not been tampered with, while simultaneously transforming the audit process into an efficient, sensible, and cost-effective procedure. Cryptography + Blockchain Hashing Process = Immutability

<u>Read this next sentence:</u>

Each transaction that is verified by the blockchain network is timestamped and embedded into a "block" of information, cryptographically secured by a hashing process that links to and incorporates the hash of the previous blockand joins the chain as the next chronological update.

In other words, if data is tampered with, the blockchain will break, and the reason could be readily identified. This characteristic is not found in traditional databases, where information can be modified or deleted with ease.

Why does immutability matter?

Complete Data Integrity—Ledgers that deploy blockchain technology can guarantee the full history and data trail of an application: once a transaction joins the blockchain, it stays there as a representation of the ledger up to that point in time. The integrity of the chain can be validated at any time by simply re-calculating the block hashes—if a discrepancy exists between block data and its corresponding hash, that means the transactions are not valid. This allows organizations and its industry regulators to quickly detect data tinkering.

Simplified Auditing—Being able to produce the complete, indisputable history of a transactional ledger allows for an easy and efficient auditing process. Proving that data has not been tampered with is a major benefit for companies that need to comply with industry regulations. Some common use cases include supply chain management, finance (for example, Sarbanes-Oxley disclosures), and identity management.

Increase in efficiencies—Maintaining a full historical record is not only a boon to auditing, but also provides new opportunities in query, analytics, and overall business processes. FlureeDB, for instance, takes advantage of the concept of time travel for business applications—where queries can be specified as of any block—or point in time—and reproduce that time's version of the database, immediately.

This capability allows for a host of time and cost savings—including tracking the provenance of major bugs, auditing specific application data, backup and restoring database state changes to retrieve information. Immutability can make the most modern-day data problems that plague enterprise applications irrelevant.

Proof of Fault—Disputes over fault in business are all-too-common. The construction industry accounts for $1 Trillion dollars in losses as a result of unresolved disputes. While blockchain won't wholly dissolve this massive category of legal proceedings, it could be leveraged to prevent a majority of disputes related to data provenance and integrity (essentially proving who did what and at what time).

Blockchain finality allows us—and a jury—to fully trust every piece of information.

Ledgers do more than just record accounting transactions. A ledger consists simply of data structured by rules. Any time we need a consensus about facts, we use a ledger. Ledgers record the facts underpinning the modern economy.

Ledgers confirm ownership.

Ledgers confirm identity.

Ledgers confirm status.

Ledgers confirm authority.

A database allows for more complex distribution, calculation, analysis and tracking. A database is computable and searchable.

But a database still relies on trust; a digitized ledger is only as reliable as the organization that maintains it (and the individuals they employ). It is this problem that the blockchain solves. The blockchain is a distributed ledgers that does not rely on a trusted central authority to maintain and validate the ledger.

Governments maintain ledgers of authority, privilege, responsibility and access. Governments are the trusted entity that keeps databases of citizenship and the right to travel, taxation obligations, social security entitlements, and property ownership. Where a ledger requires coercion in order to be enforced, the government is required. Firms and governments can use blockchains to make their work more efficient and reliable. Multinational firms and networks of firms need to reconcile transactions on a global basis and blockchains can allow them to do so near-instantaneously. Governments can use the immutability of the blockchain to guarantee that property titles and identity records are accurate and untampered. Well-designed permissioning rules on blockchain applications can give citizens and consumers more control over their data.

But blockchains also compete against firms and governments. The blockchain is an institutional technology. It is a new way to maintain a ledger—that is, coordinate economic activity [3]—distinct from firms and governments.

Distributed P2P Network [3]

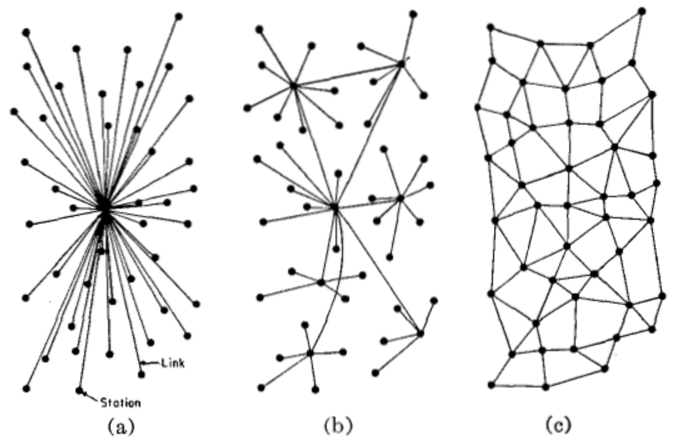


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

"Decentralization" is one of the words that is used in the cryptoeconomics space the most frequently, and is often even viewed as a blockchain's entire raison d'être, but it is also one of the words that is perhaps defined the most poorly. But there is often a lot of confusion as to what this word actually means. Consider, for example, the following completely unhelpful, but unfortunately all too common, diagram:

Blockchains are politically decentralized (no one controls them) and architecturally decentralized (no infrastructural central point of failure) but they are logically centralized (there is one commonly agreed state and the system behaves like a single computer).

Three reasons for Decentralization

The next question is, why is decentralization useful in the first place? There are generally several arguments raised:

• **Fault tolerance**— decentralized systems are less likely to fail accidentally because they rely on many separate components that are not likely.

• **Attack resistance**— decentralized systems are more expensive to attack and destroy or manipulate because they lack sensitive central points that can be attacked at much lower cost than the economic size of the surrounding system.

• **Collusion resistance**—it is much harder for participants in decentralized systems to collude to act in ways that benefit them at the expense of other participants, whereas the leaderships of corporations and governments collude in ways that benefit themselves but harm less well-coordinated citizens, customers, employees and the general public all the time.

All three arguments are important and valid, but all three arguments lead to some interesting and different conclusions once you start thinking about protocol decisions with the three individual perspectives in mind. Let us try to expand out each of these arguments one by one.

Regarding fault tolerance, the core argument is simple. What's less likely to happen: one single computer failing, or five out of ten computers all failing at the same time? The principle is uncontroversial, and is used in real life in many situations, including jet engines, backup power generators particularly in places like hospitals, military infrastructure, financial portfolio diversification, and yes, computer networks.

However, this kind of decentralization, while still effective and highly important, often turns out to be far less of a panacea than a naive mathematical model would sometimes predict. The reason is common mode failure. Sure, four jet engines are less likely to fail than one jet engine, but what if all four engines were made in the same factory, and a fault was introduced in all four by the same rogue employee?

Do blockchains as they are today manage to protect against common mode failure? Not necessarily. Consider the following scenarios:

- All nodes in a blockchain run the same client software, and this client software turns out to have a bug.

- All nodes in a blockchain run the same client software, and the development team of this software turns out to be socially corrupted.

- The research team that is proposing protocol upgrades turns out to be socially corrupted.

- In a proof of work blockchain, 70% of miners are in the same country, and the government of this country decides to seize all mining farms for national security purposes.

- The majority of mining hardware is built by the same company, and this company gets bribed or coerced into implementing a backdoor that allows this hardware to be shut down at will.

- In a proof of stake blockchain, 70% of the coins at stake are held at one exchange.

A holistic view of fault tolerance decentralization would look at all of these aspects and see how they can be minimized. Some natural conclusions that arise are fairly obvious:

- It is essential to have multiple competing implementations.

- The knowledge of the technical considerations behind protocol upgrades must be democratized, so that more people can feel comfortable participating in research discussions and criticizing protocol changes that are clearly bad.

- Core developers and researchers should be employed by multiple companies or organizations (or, alternatively, many of them can be volunteers).

- Mining algorithms should be designed in a way that minimizes the risk of centralization

- Ideally we use proof of stake to move away from hardware centralization risk entirely (though we should also be cautious of new risks that pop up due to proof of stake).

Note that the fault tolerance requirement in its naive form focuses on architectural decentralization, but once you start thinking about fault tolerance of the community that governs the protocol’s ongoing development, then political decentralization is important too.

However, once you adopt a richer economic model, and particularly one that admits the possibility of coercion (or much milder things like targeted DoS attacks against nodes), decentralization becomes more important. If you threaten one person with death, suddenly $50 million will not matter to them as much anymore. But if the $50 million is spread between ten people, then you have to threaten ten times as many people, and do it all at the same time. In general, the modern world is in many cases characterized by an attack/defense asymmetry in favor of the attacker—a building that costs $10 million to build may cost less than $100,000 to destroy, but the attacker's leverage is often sublinear: if a building that costs $10 million to build costs $100,000 to destroy, a building that costs $1 million to build may realistically cost perhaps $30,000 to destroy. Smaller gives better ratios.

What does this reasoning lead to? First of all, it pushes strongly in favor of proof of stake over proof of work, as computer hardware is easy to detect, regulate, or attack, whereas coins can be much more easily hidden (proof of stake also has strong attack resistance for other reasons). Second, it is a point in favor of having widely distributed development teams, including geographic distribution. Third, it implies that both the economic model and the fault-tolerance model need to be looked at when designing consensus protocols.

Finally, we can get to perhaps the most intricate argument of the three, collusion resistance. Collusion is difficult to define; perhaps the only truly valid way to put it is to simply say that collusion is "coordination that we don't like". There are many situations in real life where even though having perfect coordination between everyone would be ideal, one sub-group being able to coordinate while the others cannot is dangerous.

One simple example is antitrust law—deliberate regulatory barriers that get placed in order to make it more difficult for participants on one side of the marketplace to come together and act like a monopolist and get outsided profits at the expense of both the other side of the marketplace and general social welfare. Another example is rules against active coordination between candidates and super-PACs in the United States, though those have proven difficult to enforce in practice. A much smaller example is a rule in some chess tournaments preventing two players from playing many games against each other to try to raise one player's score. No matter where you look, attempts to prevent undesired coordination in sophisticated institutions are everywhere.

In the case of blockchain protocols, the mathematical and economic reasoning behind the safety of the consensus often relies crucially on the uncoordinated choice model, or the assumption that the game consists of many small actors that make decisions independently. If any one actor gets more than 1/3 of the mining power in a proof of work system, they can gain outsized profits by selfish-mining. However, can we really say that the uncoordinated choice model is realistic when 90% of the Bitcoin network's mining power is well-coordinated enough to show up together at the same conference?

The concept of a P2P distributed network. This is the physical architecture that allows Blockchain to work and provides a blockchain with redundancy.

The P2P in P2P distributed network stands for peer to peer, indicating a network comprised of peers. What do I mean by that? The majority of computer networks in place right now are what is known as Server/Client networks.

Blockchain does not use a server client approach. Instead it uses a P2p or peer to peer network to function. In a peer to peer, the nodes (laptops, tablets, etc) all talk directly to each other. Instead of a server holding all the information, the data that makes up the blockchain is instead distributed across all the different nodes. So the more nodes that are part of the blockchain, the more copies of it that exist.

This works great for redundancy as even if you took out a couple of nodes in the network, it would still be able to function as normal. And as we will see a future lesson, even if you were able to hack in and corrupt the blockchain in one of the nodes, the fact that copies of it exist on all the other nodes protect it from corruption.

This architecture is also at the heart of philosophy around crypto-currencies like BitCoin. Unlike traditional banking systems that have centralized management, BitCoin is programmed with a deflationary policy that no one person (or group of person) can control. This is an interesting economic experiment unfurling before all of us. And I for one am curious to see how it plays out. While organizations like the Fed (in the United States) have done a relatively good job of keeping the US dollar strong, poor centralized economic management has spelled disaster in countries like Venezuela and Zimbabwe.

I’ll discuss more on the economic theory behind BitCoin in later lessons. It is enough for now for you to know that the P2P decentralized nature of the network is all part of the design in ensuring no one person can make such drastic changes.

CHAPTER 4 Mining

Why is Blockchain Mining Necessary?

Blockchain Technology has prioritized fraud prevention. Any transaction will be added to the Blockchain only after it is validated. This is to prevent fake/fraud transactions. And the validation happens through Mining. So when I say you get paid for mining Blockchain, it actually means that you are getting paid for confirming the transactions.

What is Blockchain Mining?

Mining is the mechanism that allows the blockchain to be created securely and in a decentralized manner. It provides the basis for the cryptocurrency system and enables a peer-to-peer network without a central authority.

Blockchain Mining is a process used to validate new transactions. Different Blockchain implementations use different methods for validation. In this blog, I will explain an example of Bitcoin Mining. When new Blockchain transaction happens, before adding these transactions to the Block, all the miners participating in mining are given a mathematical problem. This mathematical problem is a difficult problem based on the hash algorithm which is solvable only by Brute-force.

The only way to solve this problem is to check for each possible solution to see if it is right, no shortcuts work. Finding the solution doesn't require intelligence, it just requires faster computational speed. The solution to the mathematical problem is called Proof-of-Work. The Proof-of-Work, as the name suggests is the proof that the miner has spent the time and resources to find the solution. As mentioned previously, Blockchain mining requires a lot of resources. And for spending the time and resources for this, the miner receives a reward called Mining reward.

How does Blockchain Mining work?

Blockchain Mining is mostly impossible with normal Desktop and it requires special hardware that has faster computational speed. There are two ways that mining happens: Individual Mining and Mining Pools.

Individual Mining

Here, each miner will set up the hardware and register himself for mining. When new transactions happen, all the miners in that Blockchain network receive a mathematical problem. The miners' hardware starts working on finding the solution for it. The first miner to find the solution informs all the other miners that he has found the solution. The other miners then verify it to avoid false validation of the Block. Once the solution of the miner is verified, the miner gets the reward and the transactions are added to the Blockchain.

Mining new blocks

Since there is no central authority or central bank, there bound to have a way of gathering every transaction carried out with a cryptocurrency so as to create a new block. Network nodes that perform this task are known to be dubbed "miners". Therefore, if a group of transaction is compiled into a block, it will be appended to blockchain. Hence, whoever who appends the blocks will get rewarded with some of the cryptocurrency.

Nodes

Backtracking a bit, let's talk about "nodes." A node is a powerful computer that runs the bitcoin software and helps to keep bitcoin running by participating in the relay of information. Anyone can run a node, you just download the bitcoin software (free) and leave a certain port open (the drawback is that it consumes energy and storage space – the network at time of writing takes up about 145GB). Nodes spread bitcoin transactions around the network. One node will send information to a few nodes that it knows, who will relay the information to nodes that they know, etc. That way it ends up getting around the whole network pretty quickly.

Some nodes are mining nodes (usually referred to as "miners"). These group outstanding transactions into blocks and add them to the blockchain. How do they do this? By solving a complex mathematical puzzle that is part of the bitcoin program, and including the answer in the block. The puzzle that needs solving is to find a number that, when combined with the data in the block and passed through a hash function, produces a result that is within a certain range. This is much harder than it sounds.

When the Bitcoin mining software wants to add a new block to the blockchain, this is the procedure it follows. Whenever a new block arrives, all the contents of the blocks are first hashed. If the hash is lesser than the difficulty target, then it is added to the blockchain and everyone in the community acknowledges the new block.

However, it is not as simple as that. You will have to be extremely lucky to get a new block just like that. This is where the nonce comes in. The nonce is an arbitrary string which is concatenated with the hash of the block. After that this concatenated string is hashed again and compared to the difficulty level. If it is not less than the difficulty level, then the nonce is changed and this keeps on repeating a million times until finally, the requirements are met. When that happens the block is added to the block chain.

So to recap:

- 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.

Remember property number 6 of hash functions? The puzzle friendliness?

For every output “Y”, if k is chosen from a distribution with high min-entropy it is infeasible to find an input x such that H(k|x) = Y.

So, when it comes to bitcoin mining:

- K = Nonce
- x= the hash of the block
- Y = the difficulty target

The entire process is completely random, there is no thought process behind the selection of the nonces. It is just pure brute-force where the software keep on randomly generating strings till they reach their goal. The entire process follows the Proof Of Work protocol which basically means:

- The puzzle solving should be difficult.
- Checking the answer should, however, be easy for everyone. This is done to make sure that no underhanded methods were used to solve the problem.

(For trivia lovers, this number is called a “nonce”, which is a concatenation of “number used once.” In the case of bitcoin, the nonce is an integer between 0 and 4,294,967,296.)

Solving the puzzle

How do they find this number? By guessing at random. The hash function makes it impossible to predict what the output will be. So, miners guess the mystery number and apply the hash function to the combination of that guessed number and the data in the block. The resulting hash has to start with a pre-established number of zeroes. There’s no way of knowing which number will work, because two consecutive integers will give wildly varying results. What’s more, there may be several nonces that produce the desired result, or there may be none (in which case the miners keep trying, but with a different block configuration).

The first miner to get a resulting hash within the desired range announces its victory to the rest of the network. All the other miners immediately stop work on that block and start trying to figure out the mystery number for the next one. As a reward for its work, the victorious miner gets some new bitcoin.

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.

Hashing has truly been fundamental in the creation of blockchain technology. If one wants to understand what the blockchain is all about, they should definitely understand what hashing means.

Difficulty

The difficulty of the calculation (the required number of zeroes at the beginning of the hash string) is adjusted frequently, so that it takes on average about 10 minutes to process a block.

Why 10 minutes? That is the amount of time that the bitcoin developers think is necessary for a steady and diminishing flow of new coins until the maximum number of 21 million is reached (expected some time in 2140).

If you’ve made it this far, then congratulations! There is still so much more to explain about the system, but at least now you have an idea of the broad outline of the genius of the programming and the concept. For the first time we have a system that allows for convenient digital transfers in a decentralized, trust-free and tamper-proof way. The repercussions could be huge.

CHAPTER 5 Consensus

Consensus Protocol

Consensus protocols are one of the most important and revolutionary aspects of blockchain technology.

These protocols create an irrefutable system of agreement between various devices across a distributed network, whilst preventing exploitation of the system.

Blockchain consensus protocols are the hub players controlling the nodes on a network synchronized with each other, while providing an answer to the question: <u>how do we all make sure that we agree on what the truth is?</u>

After all, anyone can submit information to be stored onto a blockchain and therefore it is important that there is review and confirmation, in the form of a consensus about whether to add that information.

What is a protocol?

A set of rules describing how the communication and transmitting of data between electronic devices, such as nodes, should work. These rules need to be defined before any data is sent, detailing how the information will be structured and how each device will send or receive it. [4]

As a term, **‘consensus’** means that the nodes on the network *agree* on the same state of a blockchain, in a sense making it a self-auditing ecosystem. This is an absolutely crucial aspect of the technology, carrying out two key functions. <u>Firstly</u>, consensus protocols allow a blockchain to be updated, while ensuring that every block in the chain is true as well as keeping participants incentivized. <u>Secondly</u>, it prevents any single entity from controlling or derailing the whole blockchain system. The aim of consensus rules is to guarantee a single chain is used and followed.

Consensus Protocol Rules

Consensus rules are a specific set of rules that nodes on the network will ensure a block follows when validating that block and the transactions within it. The key requirement to achieve a consensus is a unanimous acceptance between nodes on the network for a single data value, even in the event of some of the nodes failing or being unreliable in any way.

Every cryptocurrency must have a way of securing its blockchain against attacks. For example, an attacker may attempt to spend some money and then reverse the transaction by broadcasting their own version of that blockchain, not including the transaction. This is known as a double spend. As blockchain technology does not rely on a central authority for security, users have no prior knowledge which version of the record is valid.

Consensus protocols are designed to be difficult to imitate or replicate by being extremely costly to carry out, in terms of time, the computing resources required or the holdings of a particular cryptocurrency. The methods of consensus vary depending on the blockchain within which they are validating the blocks and there exist a variety of forms of consensus, with a consistent ongoing debate as to what is the most effective and efficient method.

Consensus protocols are a key aspect in allowing a blockchain to function and exist. After all, as at its core a blockchain is a ledger of information it is paramount that there is absolute certainty that the information that is being stored is honest and accurate.

We hear plenty of talk of how public blockchains are going to change the world, but to function on a global scale, a shared public ledger needs a functional, efficient and secure consensus algorithm.

A consensus algorithm, like bitcoin's proof of work (the one we hear about most often), does two things: it ensures that the next block in a blockchain is the one and only version of the truth, and it keeps powerful adversaries from derailing the system and successfully forking the chain.

In proof of work, miners compete to add the next block (a set of transactions) in the chain by racing to solve a extremely difficult cryptographic puzzle. The first to solve the puzzle, wins the lottery. As a reward for his or her efforts, the miner receives 12.5 newly minted bitcoins – and a small transaction fee.

While not a comprehensive list, the following are a few of the alternative approaches being kicked around out there.

Proof of Work (PoW)

Creating the proof of work protocol for achieving consensus between devices on a distributed network is arguably the crowning achievement of Bitcoin founder Satoshi Nakamoto. In doing so, he laid the groundwork for the revolutionary technology that is blockchain.

Proof of work (abbreviated to PoW) is a consensus protocol introduced by Bitcoin and used widely by many other cryptocurrencies. This process is known as mining and as such the nodes on the network are known as "miners". The "proof of work" comes in the form of an answer to a mathematical problem, one that requires considerable work to arrive at, but is easily verified to be correct once the answer has been reached. In Proof of Work, in order for an actor to be elected as a leader and choose the next block to be added to the blockchain they have to find a solution to a particular mathematical problem. This is the most popular algorithm being used by currencies such as Bitcoin and Ethereum, each one with its own differences.

The only way to solve these mathematical riddles is through nodes on the network, running a long and random process of presenting answers on a trial and error basis. Technically, this means that the problem could be solved on first attempt, although this is extremely unlikely, to the point where it is practically impossible. The answer needs to be a lower number than the hash of the block for it to be accepted, known as the 'target hash'.

A target hash is a number that the header of a hashed block must be equal to or less than for a new block, along with the reward, to be awarded to a miner. The lower a target is, the more difficult it is to generate a block. A miner continues testing different unique values (known as nonces) until a suitable one is produced. The miner who manages to solve the riddle mines the next block, adding it to the chain and validating the transactions within it, and receiving the reward associated with the block.

The process involves ensuring every confirmed block in the chain rewards the miner in the cryptocurrency that they are mining through the transaction fees collected for sending currency across the network, as well as any predetermined reward. It ensures that miners are incentivized to continue maintaining a blockchain, as they are being rewarded for doing so.

These rewards are especially important due to the complexity of the riddles that are being solved since the process is extremely costly, both in the terms of time taken and the computing power required to do so. Keeping these miners incentivized is a key function of a protocol as they are in a sense the foundation that keep the system running. Systems such as proof of work are employed so transactions cannot be counterfeited, as the data required to do so is extremely difficult to produce, yet easily verified.

Proof of Stake (PoS)

Proof of stake is the consensus algorithm used by cryptocurrencies to validate blocks. The most common alternative to proof of work is proof of stake. The main advantages of proof of stake are energy efficiency and security.

In this type of consensus algorithm, instead of investing in expensive computer equipment in a race to mine blocks, a ‘validator’ invests in the coins of the system.

Note the term validator. That’s because no coin creation (mining) exists in proof of stake. Instead, all the coins exist from day one, and validators (also called stakeholders, because they hold a stake in the system) are paid strictly in transaction fees.

In proof of stake, your chance of being picked to create the next block depends on the fraction of coins in the system you own (or set aside for staking). A validator with 300 coins will be three times as likely to be chosen as someone with 100 coins.

The randomization in a proof of stake system prevents centralization, otherwise the richest individual in the system would always be creating the next block and consistently increasing their wealth and as a result their control of the system. The main advantage of proof of stake, over a system such as proof of work, is that it uses considerably less energy and as a result is more cost effective. It is well documented that each Bitcoin transaction, which uses a proof of work system, can require as much electricity as an average Dutch household does in two weeks. This is both ineffective and unsustainable.

In that regard proof of stake can be regarded as a superior consensus protocol as it requires far less electricity to run. Furthermore, as the proof of stake system is so much more cost effective there is less of a need to release too many new coins as a means of incentivizing miners to maintain the network. This helps to keep the price of a particular coin more stable.

Proof of stake protocol is effective in not only encouraging individuals to partake in the system but also preventing any individual from controlling the network. In order to carry out a 51% attack an individual or group would need to own the majority of coins on the network.

Firstly, it would be extremely expensive to acquire enough coins to get anywhere near doing so since many individuals would likely exit the currency if a single party began buying everything, while others would ramp up the price to discourage a hostile takeover. Furthermore, it would be completely counterproductive to attack the network as it would vastly decrease the value of the coins that the attacker is holding. Essentially, the users with the highest stake in a cryptocurrency have the most interest maintaining and securing the network because any attacks would diminish the reputation and price of the cryptocurrency that they hold.

In Proof of Work, if Bob has more computational power and energy than Alice—and thus can output more work—he is more likely to win (mine the next block).

Similarly, yet again:

In Proof of Stake, if Bob has more stake than Alice, he is more likely to win (“mine” the next block).

Proof of Stake takes away the energy and computational power requirement of PoW and replaces it with stake. Stake is referred to as an amount of currency that an actor is willing to lock up for a certain amount of time. In return, they get a chance proportional to their stake to be the next leader and select the next block.

However, proof of stake does have its downsides, one of them being a “nothing at stake” problem. The issue occurs in the event of a consensus failure when block-generators have nothing to lose by supporting varying blockchain histories, preventing the conflict from resolving.

Overall, the proof of stake consensus protocol is a robust system that effectively and efficiently fulfills its intended purpose. However, this has not stopped companies from modifying and improving the protocol, an example of this being Delegated Proof of Stake, otherwise known as DPoS.

Whereas in a proof of work system, such as the one employed by Bitcoin, validating blocks is known as “mining”, in the case of delegated proof of stake this process is referred to as “forging”.

The basis of how **consensus** is agreed upon for each transaction added to the ledger.

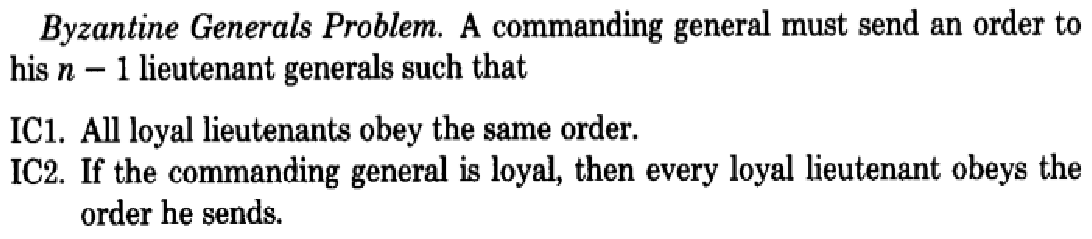
What are the benefits of each type of consensus mechanism and in which situation are they best utilized?

Proof of Work—Miners have a financial incentive to process as many transactions as quickly as possible. PoW is best utilized by high-throughput requirement systems.

Proof of Stake—Transaction Validators receive rewards in proportion to the amount of their “stake” in the network. This arguably improves network security by discouraging duplicitous attacks. PoS is best used by computing power constrained organizations.

CHAPTER 6 The Byzantine Generals Problem

Famously described in 1982 by Lamport, Shostak and Pease [5], it is a generalized version of the Two Generals Problem with a twist. The Byzantine General's Problem is one of many in the field of agreement protocols. It describes the same scenario, where instead more than two generals need to agree on a time to attack their common enemy. Two generals have to come to a common agreement on whether to attack or retreat, but can communicate only by sending messengers who might never arrive. The extra complication here is that one or more of the generals can be a traitor, meaning that they can lie about their choice (e.g. they say that they agree to attack at 0900 but instead they do not).



Byzantine Generals Problem. **A commanding general must send an order to his $n - 1$ lieutenant generals such that**

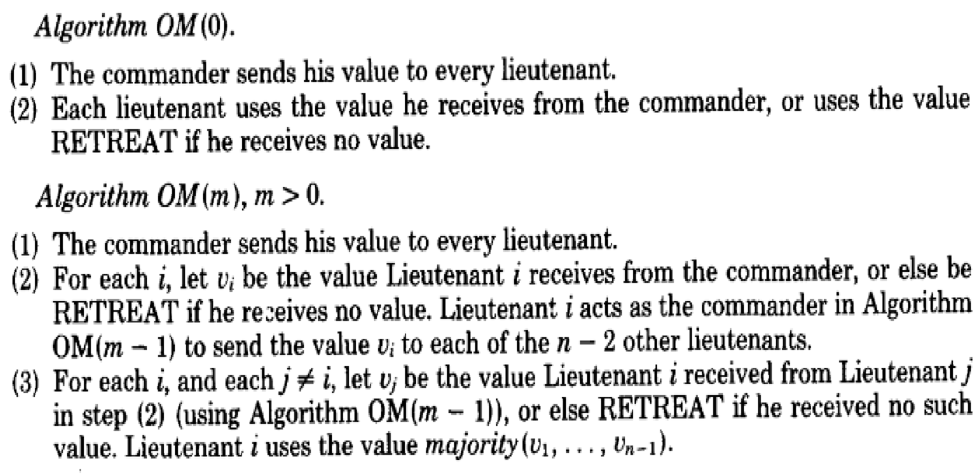
IC1. All loyal lieutenants obey the same order.
IC2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

Adding to IC2., it gets interesting that if the commander is a traitor, consensus must still be achieved. As a result, all lieutenants take the majority vote.

The algorithm to reach consensus in this case is based on the value of majority of the decisions a lieutenant observes.

Theorem: For any m, Algorithm OM(m) reaches consensus if there are more than 3m generals and at most m traitors.

This implies that the algorithm can reach consensus as long as 2/3 of the actors are honest. If the traitors are more than 1/3, consensus is not reached, the armies do not coordinate their attack and the enemy wins.



Algorithm OM(0).

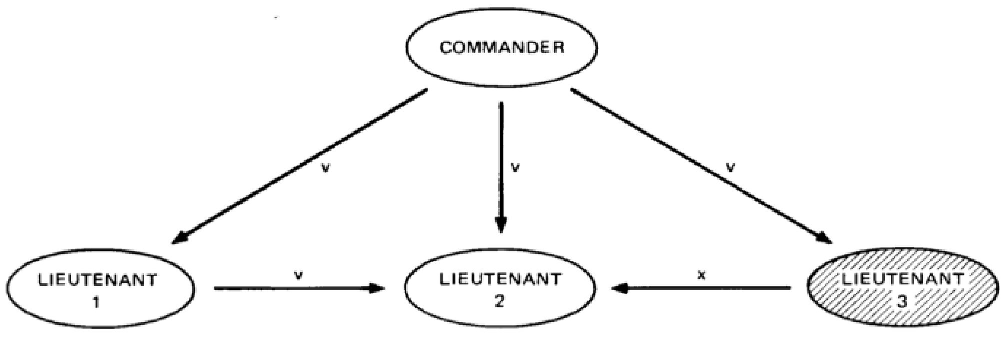
(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Algorithm OM(m), $m > 0$.

(1) The commander sends his value to every lieutenant.
(2) For each i, let v_i be the value Lieutenant i receives from the commander, or else be RETREAT if he receives no value. Lieutenant i acts as the commander in Algorithm OM($m - 1$) to send the value v_i to each of the $n - 2$ other lieutenants.
(3) For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (using Algorithm OM($m - 1$)), or else RETREAT if he received no such value. Lieutenant i uses the value *majority*($v_1, \ldots, v_{n-1}$).

m = 0 → no traitors, each lieutenant obeys | m > 0 → each lieutenant's final choice comes from the majority of all lieutenant's choices

This should be more clear with a visual example from Lieutentant 2's point of view— Let C be Commander and L{i} be Lieutenant i:



OM(1): Lieutenant 3 is a traitor—L2 point of view

Steps:

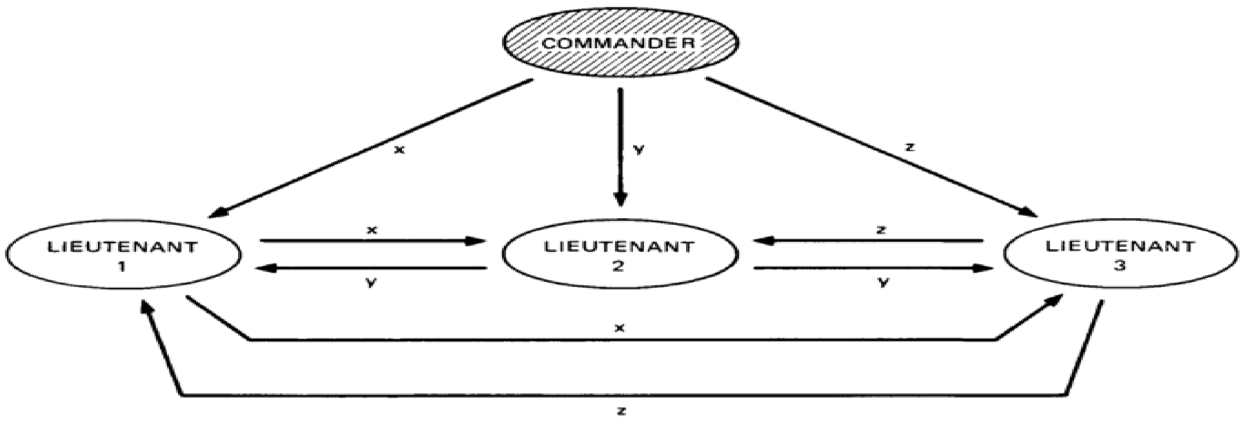
Commander sends v to all Lieutenants

L1 sends v to L2 | L3 sends x to L2

L2 ← majority(v,v,x) == v

The final decision is the majority vote from L1, L2, L3 and as a result consensus has been achieved

The important thing to remember is that the goal is for the majority of the lieutenants to choose the same decision, not a specific one. Let’s examine the case of *the commander being a traitor:*



OM(1): Commander is a traitor

Steps:

Commander sends x, y, z to L1, L2, L3 respectively

L1 sends x to L2, L3 | L2 sends y to L1, L3 | L3 sends z to L1, L2

L1 ← majority(x,y,z) | L2 ← majority(x,y,z) | L3 ← majority(x,y,z)

They all have the same value and thus consensus is reached. Take a moment here to reflect that even if x, y, z are all different the value of majority(x, y, z)is the same for all 3 Lieutenants. In the case x,y,z are totally different commands, we can assume that they act on the default option retreat.

The leader-follower paradigm described in the Two Generals Problem is transformed to a commander-lieutenant setup. In order to achieve consensus here, the commander and every lieutenant must agree on the same decision (for simplicity attack or retreat).

In a few words, Byzantine fault tolerance (BFT) is the property of a system that is able to resist the class of failures derived from the Byzantine Generals’ Problem. This means that a BFT system is able to continue operating even if some of the nodes fail or act maliciously.

There is more than one possible solution to the Byzantine Generals' Problem and, therefore, multiple ways of building a BFT system. Likewise, there are different approaches for a blockchain to achieve Byzantine fault tolerance and this leads us to the so-called consensus algorithms.

How does this all relate to blockchain?

Blockchains are decentralized ledgers which, by definition, are not controlled by a central authority. Due to the value stored in these ledgers, bad actors have huge economic incentives to try and cause faults. That said, Byzantine Fault Tolerance, and thus a solution to the Byzantine Generals' Problem for blockchains is much needed.

In the absence of Byzantine Fault Tolerance, a peer is able to transmit and post false transactions effectively nullifying the blockchain's reliability. To make things worse, there is no central authority to take over and repair the damage.

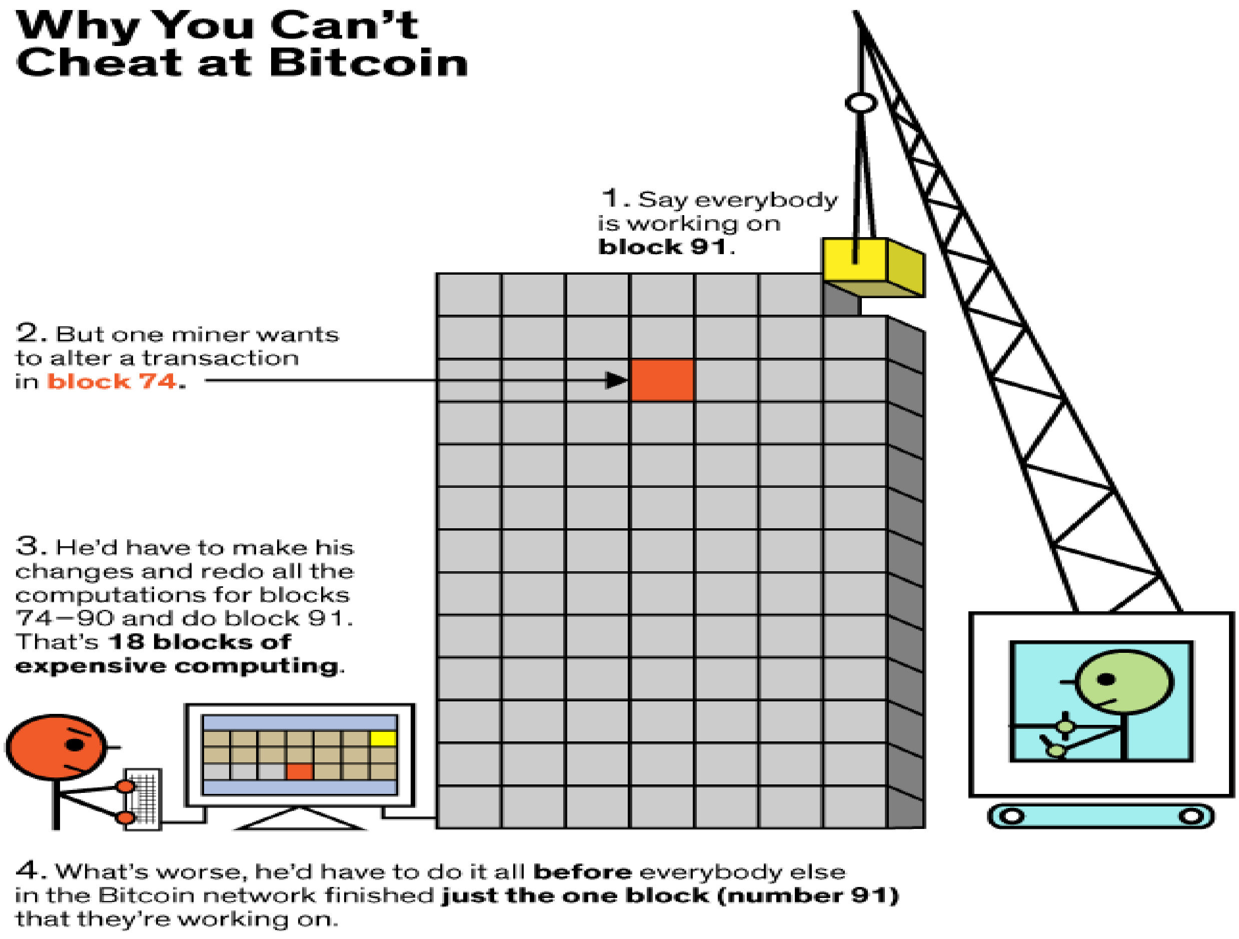
Here , it comes as a result , that we have to understand how we can solve the Byzantine Fault Tolerence , due to the absence of a third trustful party , which is the Consensus we have already discussed at the previous chapter.

Consensus

Are there alternative algorithms which achieve Byzantine Fault Tolerance?

Proof of Work (PoW)

In regards to the issue of Byzantine Fault Tolerance, the proof of work protocol deals with the problem of Byzantine nodes through nonces and combining messages into blocks. Each block has its own distinct nonce. They are only used once in order to add another element of difficulty in generating valid hashes, specifically to prevent precomputation and ensure fairness. Despite having some merits, proof of work is regarded as a flawed consensus protocol, especially when considering how much energy is consumed in running the protocol.


Why You Can't Cheat at Bitcoin
1. Say everybody is working on **block 91**.
2. But one miner wants to alter a transaction in **block 74**.
3. He'd have to make his changes and redo all the computations for blocks 74–90 and do block 91. That's **18 blocks of expensive computing**.
4. What's worse, he'd have to do it all **before** everybody else in the Bitcoin network finished **just the one block (number 91)** that they're working on.

The Byzantine Generals' Problem is an intriguing dilemma that eventually gave rise to the Byzantine Fault Tolerance (BFT) systems, which are being extensively applied in various scenarios. Beyond the blockchain industry, a few use cases of Byzantine Fault Tolerance (BFT) systems include the aviation, space, and nuclear power industries.

Within the cryptocurrency context, having an efficient network communication along with a good consensus mechanism is vital to any blockchain ecosystem. Securing these systems is an ongoing effort, and the existing consensus algorithms are yet to overcome a few limitations (such as scalability). Nonetheless, PoW and PoS are very interesting approaches as Byzantine Fault Tolerance (BFT) systems, and the potential applications are certainly inspiring widespread innovation.

CHAPTER 7 Cryptocurrency - mempool

Mining Pool

How do Mempools Work?

The mempool is the node's holding area for all the pending transactions. It is the node's collection of all the unconfirmed transactions it has already seen enabling it to decide whether or not to relay a new transaction.

There are as many mempools as there are as nodes.

As the Bitcoin network is distributed, not all nodes receive the same transactions at the same time so some nodes store more transactions than others at some time. Plus, everyone can run its own node with the hardware of his choice; so all nodes have a different RAM capacity to store unconfirmed transactions. As a result, each node has its own rendition of the pending transactions, this explains the variety of Mempool sizes & transactions counts found on different sources.

How do transactions get into the Mempool? [6]

Before letting a transaction into its Mempool, a node has to complete the series of checks listed below.

- Check syntactic correctness
- Make sure neither in or out lists are empty
- Size in bytes < MAX_BLOCK_SIZE
- Each output value, as well as the total, must be in legal money range
- Make sure none of the inputs have hash=0, n=-1 (coinbase transactions)
- Check that nLockTime <= INT_MAX, size in bytes >= 100, and sig opcount <= 2
- Reject "nonstandard" transactions: scriptSig doing anything other than pushing numbers on the stack, or scriptPubkey not matching the two usual forms

- Reject if we already have matching tx in the pool, or in a block in the main branch
- For each input, if the referenced output exists in any other tx in the pool, reject this transaction.
- For each input, look in the main branch and the transaction pool to find the referenced output transaction. If the output transaction is missing for any input, this will be an orphan transaction. Add to the orphan transactions, if a matching transaction is not in there already.
- For each input, if the referenced output transaction is coinbase (i.e. only 1 input, with hash=0, n=-1), it must have at least COINBASE_MATURITY (100) confirmations; else reject this transaction
- For each input, if the referenced output does not exist (e.g. never existed or has already been spent), reject this transaction Using the referenced output transactions to get input values, check that each input value, as well as the sum, are in legal money range
- Reject if the sum of input values < sum of output values
- Reject if transaction fee (defined as sum of input values minus sum of output values) would be too low to get into an empty block
- Verify the scriptPubKey accepts for each input; reject if any are bad
- Add to transaction pool
- “Add to wallet if mine”
- Relay transaction to peers
- For each orphan transaction that uses this one as one of its inputs, run all these steps (including this one) recursively on that orphan.

More simply, the users add new transactions. So, (pre-verified transactions) are inserted to the mempool. Mempool , stores the transactions until a given miner verifies the transactions and inserts them to the block . Miners take the transactions and put them into the blocks.

Miners select M transactions from the Mempool and create a new block in the blockchain with these transactions and also, they find the right hash for the block. Every transaction has a transaction fee which is the amount the user is willing to pay for making the given transaction, which will be the reward of the miner. It is basically an optimization problem (bin-packing): we have a bin capacity C (1MB) and we have N items with values $t_1,\ t_2,\ t_3$.

What items to include in the bin to maximize the profit?

Miners will sort items and take the ones with the highest transaction fees (note the miners get the sum of fees after mining).

How do miners selct the optimal set of transactions?

Miner’s reward = X BTC + Transactions Fees.

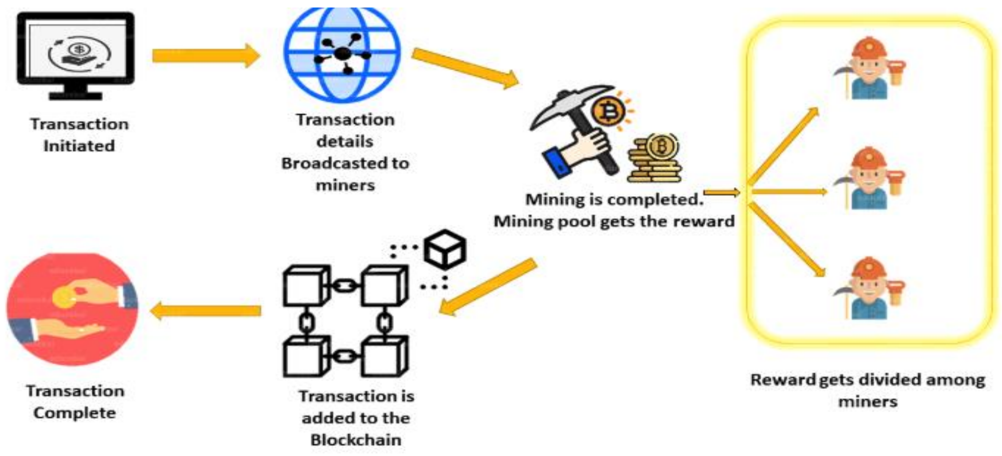
If the transaction matches the above criteria, it is allowed into the Mempool and the node starts broadcasting it. If it doesn’t the transaction is not re-broadcast by the node.

How does a new block impact the Mempool?

When a node receives a new valid block, it removes all the transactions contained in this block from its mempool as well as the transactions that have conflicting inputs.

What happens when the node’s memory get full?

Unlike mining, there is no financial incentive for running a node. Therefore, the hardware dedicated to it tends to be limited and so a node’s Mempool often max out its RAM. When this happen, in former versions of bitcoind, the node would just crash and restart with an empty Mempool.


Transaction Initiated
Transaction details Broadcasted to miners
Mining is completed. Mining pool gets the reward
Reward gets divided among miners
Transaction is added to the Blockchain
Transaction Complete

Picture : Role of Mining Pool and of Miners

CHAPTER 8 Merkle Tree

Merkle Tree - Cryptographic proof which transaction are in the block

What is a Merkle Tree?

Merkle trees are a data structure involving hashes useful for efficient data verification - essentially allowing one data point to be verified and in so doing affirming the validity of multiple data points. The trees are constructed by pairwise (or n-wise) grouping of a dataset and sequential hashing.

Put simply, a Merkle Tree, also called a binary tree, is a method of structuring data that enables quick and efficient verification of huge chunks of information.

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. Doing so will make finding any particular transaction extremely cumbersome and time-consuming. If you use a Merkle tree, however, you will greatly cut down the time required to find out whether a particular transaction belongs in that block or not.

Now suppose I want to find out whether this particular data belongs in the block or not: Instead of going through the cumbersome process of looking at each individual hash and seeing whether it belongs to the data or not, I can simply track it down by following the trail of hashes leading up to the data: Doing this significantly reduces the time taken.

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.

In order to understand more in depth the functionality of Merkle Tree , let’s consider why we use SHA256.

Why we use SHA256?

We use SHA256 hashes to identify a given block in the blockchain. It is not that optimal to include all the transactions in the header because there can be 100-800 transactions within a single block and we want to represent all these transactions with a single hash.

What does Merkle Tree do ?

- There is a tree-like structure and store the root of this tree in the header (entries are SHA256 hashes)

- This Merkle-root can verify all the transaction and it is just a single hash value.

If any of the transactions in the block changes then the root’s value is changed as well: so this is why we can verify all the transactions with the Merkle-root exclusively.

We represent a set of transactions with the help of a SHA256 so a 64 characters long hexadecimal string (the Merkle-root itself). If a single detail in any of the transactions changes or even the order of the transactions then the Merkle-root will change as well
It is part of the block’s header which forms the hash of the block (after applying the SHA256 algorithm). *Merkle Root is the cryptographic proof of which transactions are in the block.*

- **without the Merkle-root in the block’s header**: we would not have proof of which transactions are included in the given block and that their contents have not been tampered with.

CHAPTER 9 Elliptic Curve Cryptography

There is a huge problem, all the data is public. Somehow, we have to encrypt the transactions and have to make sure that other nodes in the network can verify these transactions. For this reason, Bitcoin uses ECDSA (Elliptic Curve Digital Signature Algorithm) to ensure that funds can only be spent by their rightful owners.

Compared to RSA encryption, Elliptic Curve Cryptography offers a significant advantage. The key size used for Elliptic Curve Cryptography is much smaller than what is needed for RSA encryption, while still providing the same level of security. Although RSA encryption is more widely used across the Internet today, Elliptic Curve Cryptography is essentially a more efficient form of RSA, which is one of the primary reasons it is used in cryptocurrencies. As an example of the efficiency of Elliptic Curve Cryptography as compared to RSA, the same 384-bit key used in encrypting classified information would require a 7680-bit key using RSA encryption. The efficiency afforded by Elliptic Curve Cryptography is therefore exceedingly useful to blockchain networks since it reduces the size of transactions.

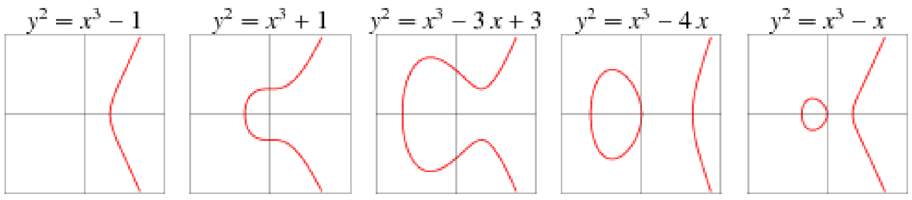
The ***Elliptic Curve Digital Signature Algorithm (ECDSA****), defines a technique for generatingand* ***validating digital signatures****.*

Elliptic Curve Cryptography is a method of public-key encryption based on the algebraic function and structure of a curve over a finite graph. It uses a trapdoor function predicated on the infeasibility of determining the discrete logarithm of a random elliptic curve element that has a publicly known base point.

Trapdoor functions are used in public-key cryptography to make it so, going from A—> B is trivial, but going from B —> A is infeasible, by leveraging a specific mathematical problem. For instance, RSA encryption is based on the concept of Prime Factorization, and Elliptic Curve Cryptography relies on the concept of Point Multiplication, where the multiplicand represents the private key and is infeasible to compute from the given starting points.
An elliptic curve consists of all the points that satisfy an equation of the following form:

$y^2 = x^3+ax+b$,where $4a^3+27b^2 \neq 0$ (this is required to avoid singular points).

Here are some example elliptic curves [7]:



Notice that all the elliptic curves above are symmetrical about the x-axis. This is true for every elliptic curve because the equation for an elliptic curve is:

$y^2 = x^3+ax+b$.

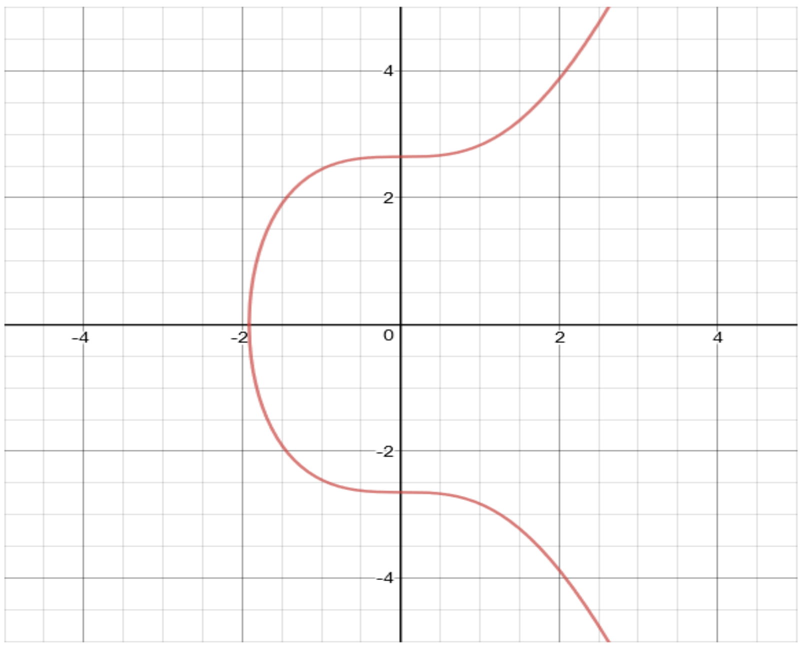
And if you take the square root of both sides you get:

$y = \pm\sqrt{x^3 + ax + b}$,

So, if a = 27 and b = 2 and you plug in x = 2, you'll get y = ±8, resulting in the points (2, -8) and (2, 8).

The elliptic curve used by Bitcoin, Ethereum, and many other cryptocurrencies is called secp256k1 [8].

The equation for the secp256k1 curve is $y^2 = x^3+7$. This curve looks like:



<u>Private and public keys in elliptic curve cryptography</u>

Let’s say I compute x•P, where x is a random 256-bit integer. The result will be some point on the curve. Let’s call that point X.

If I give you X, could you determine x? In other words, could you determine how many times I added P to itself to get the point X on the curve? Let’s assume that you know what P is and you know what curve I was using.

It turns out that is not feasible for you to figure out x, even if you had a super computer. There is no known algorithm for determining x, so your only option is to keep adding P to itself until you get X or keep subtracting P from X until you get P. On average, x will be somewhere between 0 and 2^{256}-1, about 2^{128}. Thus, it will take you on average 2^{128} point addition operations to determine x no matter your approach. Even if your computer could do one trillion point addition operations per second and you had been running your computer since the beginning of the universe, you would’ve only done 2^{98} point addition operations by now. $2^{98}/2^{128}$ =1/1073741824.

<u>What if you start in the middle?</u> You can calculate 2^{128}•P in 510 steps or less after all. Well, on average, x is no closer to 2^{128} than it is to 0 or 2^{256}-1, because x is random, so it doesn’t matter where you start—you will still have to do 2^{128} point addition operations on average.

Private and public keys

So, because someone can’t figure out x given X, where X=x•P, it might be convenient to make x your private key and X your public key. Your private key would then be a random 256-bit integer and your public key would be the x- and y- coordinates of a point on an elliptic curve. This would satisfy the following property of private and public keys:
“It is computationally infeasible to derive the private key corresponding to a given public key.”

Furthermore, while we haven’t discussed this yet, it is possible to prove to someone that you know x, without revealing any useful information about x. That is, you can show someone that you know how many times you would have to add P to itself to get X without straight up telling them what x is.

How to prove you know x?

So if you compute X=x•P, where x is a random 256-bit integer, how can you prove to someone that you know the x that corresponds to X without revealing any useful information about x?

You can use the point addition property from earlier:

n•P + r•P = (n+r)•P .

We will modify it slightly:

hash(m, r•P)•n•P+r•P = (hash(m, r•P)*n+r)•P .

If you expand the right-hand side of the equation above, you will get the left-hand side, so the equation above holds for any m, r, and n.

So what happens if we set n•P=X? We'll have:

hash(m, r•P)•X+r•P = (hash(m, r•P)*n+r)•P

If n•P = X, then n = x, so we have:

hash(m, r•P)•X+r•P = (hash(m, r•P)*x+r)•P

Now we're going to make the following substitutions: R = r•P and s = hash(m,R)*x+r.

So now we have:

hash(m, R)•X+R = s•P

Okay, here's the claim: if you can provide an m, R, and s that satisfy the above equation, then this proves that you know the x corresponding to the X in the equation, where x•P =X.

For this to be the case, the following two conditions must be met:

1. If you know x, then you should be able to provide working values for m, R, and s.
2. If you don't know x, then you should not be able to provide working values for m, R, and s.

If you know x, you can clearly come up with working values for m, R, and s. Choose random values for m and r, then compute R=r•P and s=hash(m)*x+r. If you plug these values into hash(m,R)•X+R=s•P, then you get:

hash(m, r•P)•x•P+r•P = (hash(m, r•P)*x+r)•P

which is the equation that we said would hold earlier for any m, r, and n (x is n in this case).

What if you don't know x? Could you come up with working values for m, R, and s? The problem is that you would have to solve hash(m,R)•X+R = s•P. Basically, you would have to find an input for a hash that has a specific hash value, which is not possible, or at least it's not computationally feasible, because of the preimage resistance property of cryptographic hash functions.

Therefore, the only way to provide working values for m, R, and s is by computing them using x. Thus, you can prove that you know the private key, x, that goes with the public key, X, by providing values for m, R, and s that satisfy hash(m,R)•X+R=s•P.

Do you reveal any useful information about x by proving that you know x?

If you provide working values for m, R, and s, can anything useful about x be gleaned from those values?
m and R have nothing to do with x, so those values can't reveal any useful about x.
We know that s = hash(m,R)*x+r. Could someone compute x from s?
To do so, they would have to solve x = (s-r)/hash(m, R).
Since they don't know r, they can't compute x from s. They can't obtain r from the fact that R = r•P ,(they are given R), because that's the same as determining how many times P would have to be added to itself to get R, which is the same computationally infeasible problem that prevents someone from determining x from X.
Also , it doesn't reveal any information about x, such as "x must be less than yadda yadda". If r is generated randomly and we allow hash(m, R)*x+r to overflow so r can be any 256-bit integer, then the value for s is completely random, meaning s could be any 256-bit integer. A random 256-bit integer tells you just as much as about x as the GDP of New Zealand does.

Digital signatures [9]

The variables m, R, and s can be used to prove that one knows the x that corresponds to X, where X = x•P. Verifying the proof requires plugging m, R, and s into hash(m,R)•X+R = s•P. Can we make it so that a specific message is required for the verification to be successful, so that the proof—m, R, and s—forms a digital signature for that message? Yes!

Let m be that specific message and R and s be the digital signature for that message. The verification process would then only be successful if the specific message, m, is plugged into the verification equation. If a different value for m is plugged in, then the left-hand side of hash(m,R)•X+R = s•P , would fail to equal the right-hand side, because s was calculated using a different message.

Thus, one can prove that they know the private key, x, that corresponds to a public key, X, for a specific message, m, by providing a digital signature R and s for m.

For cryptocurrencies, the message would be the unsigned part of a transaction. If you ever look at a digital signature for a transaction, it is generally the x-coordinate of R (R is a point on the curve) concatenated with s (s is a seemingly random 256-bit integer), after it has been encoded and converted to hexadecimal.

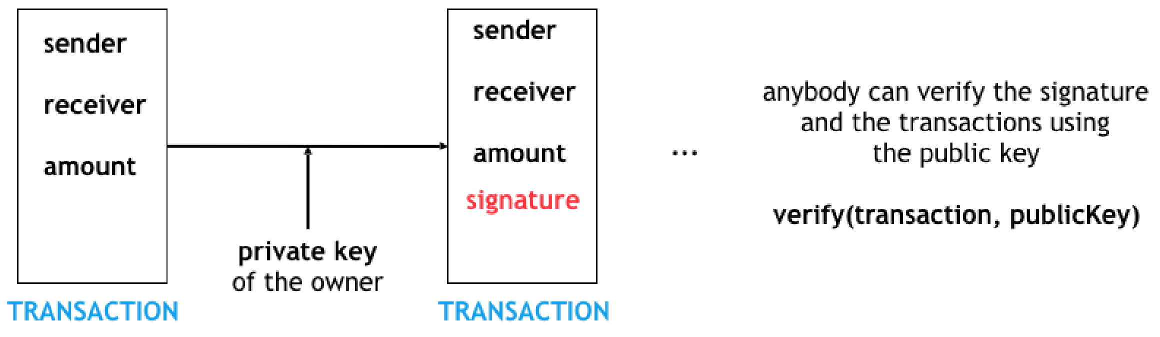
If you want to obtain a Bitcoin address or Ethereum account, you generate a random 256-bit integer x. x is then your private key. Then you compute X = x•P using the parameters for the secp256k1 curve. X will be your public key. Your public key is safe to give out and cannot be used to determine your private key. If you hash your public key, you will obtain your address.

When you want to send bitcoin or ether from your address to another address, you create a transaction. You set m to the unsigned part of that transaction, and compute R and s from that m. Then you attach R and s to the transaction. After you broadcast your transaction, any node will be able to verify that m (the unsigned part of the transaction), R, and s satisfy hash(m,R)•X+R = s•P. This of course assumes that you also include X in your transaction, since your public key cannot be determined from your address. For Ethereum, instead of providing X you provide v, which allows one to determine X from R and s.

To simplify all of the above and make them easier to understand them:

PRIVATE KEY : it is a secret number (256 bits integer) known only to the person that generated it. With this private key a transaction can be signed.

PUBLIC KEY : it is generated from the private key and no need to keep it secret. It is extremely hard to get the private key from the public key (public key is a 2D point coordinate on an elliptic curve). The public key , can help to verify the message (that has the signature).


sender
receiver
amount
TRANSACTION
private key
of the owner
sender
receiver
amount
signature
TRANSACTION
...
anybody can verify the signature and the transactions using the public key
verify(transaction, publicKey)

Use In Cryptocurrencies

Aiming to analyse the need for digital signatures schemes in cryptocurrencies, there are four primary requirements of any given scheme that must be met for the signature scheme to be provably authentic and verifiable. These include:

1. It should be provably verifiable that a signer of a transaction is the signer.

2. The signature should not be forgeable.

3. The signature needs to be non-reputable, meaning signatures are final and cannot be associated with another identity.

4. It should be computationally infeasible to derive the private key from a corresponding public key.

Elliptic Curve Cryptography satisfies all 4 conditions and is also particularly effective in doing so. Using Elliptic Curve Cryptography, the (x, y) coordinates of a point on the graph would be your public key, and the 384-bit random integer x would be your private key.

It is also possible to prove to somebody that you know the value of x, without actually revealing what x is. This property further helps to satisfy the necessary conditions for sustainable use in a digital signature transaction scheme.

For now, Elliptic Curve Cryptography and other digital signature schemes utilizing trapdoor functions remain some of the most secure encryption methods in the world and should continue to remain so for some time.

CHAPTER 10 Unspent Transactions Outputs - UTXOs

What Is a UTXO?

Bitcoin, and many protocols based on it, store data about transactions and user balances in the form of unspent transaction outputs, which are a list of "unspent" Bitcoin amounts that have been sent *to* a user, but have not yet been sent *from* him/her. The sum of these outputs is the user's total balance. On the blockchain, they appear to be a collection of Bitcoin amounts on different addresses, and the role of a wallet is to identify which addresses the user has keys to. Individual Bitcoin are easy to track because they are signed from one person to another. A transaction is valid if one can prove ownership over the actual Bitcoin s/he is trying to send. [10]

Unspent Transaction Outputs (UTXO) are all the spendable outputs for a given address available to be used in new transactions. There's a conceptual dependency here which is understanding how transactions work. Let's use Bitcoin to analyze this.

A Bitcoin transaction is comprised of inputs and outputs. Only Unspent Transaction Outputs, or UTXOs, can be used to be spent as an input in another transaction whereas spent outputs are already spent hence can't be spent again. You always need a UTXO or an unspent transaction output to make a transaction. If you don't have an unspent transaction output, it simply means you don't have any Bitcoin.

This mainly happens due to the protocol rules which Satoshi Nakamoto [11] had defined in Bitcoin to prevent double spending. Also, there is no way in the bitcoin world to spend partial amounts while completing a transaction.

To break it down further: If you have a balance of 3 BTC on 'XYZ' public address and you have to pay 0.5 BTC to a merchant, you cannot simply send 0.5 out of your 'XYZ' address and keep the rest 2.5 BTC intact.

Instead, you need to spend whole 3 BTC out of which you will designate 0.5 BTC to the merchant while providing a signature and sending the rest 2.5 BTC back to yourself on an address that you control. This is called sending the change to the change address. You might be wondering that you never make two transactions in your wallet when you pay someone. Yes, that's true because modern wallets take care of everything behind the scenes.

In short, when a bitcoin transaction takes place, there are two UTXOs created: one that is the actual coins sent to the recipient, and one that is the change output, which goes back to the sender's wallet.

To quickly summarize, all transactions have both inputs and outputs. Inputs are where the money came from and outputs are where the money is going. Unless you're a miner receiving newly created Bitcoin, every output sent to you will contain one or more inputs indicating where the money originated from. If you keep following this trail back you'll eventually arrive at the the first transaction of a given block—the coinbase transaction.
So all Bitcoin transactions therefore have **the following constraints**:

1. Every input must be valid and not yet spent
2. The transaction must have a signature matching the owner of the input
3. The total value of the inputs must equal or exceed the total value of the outputs

In the case of a valid blockchain transaction, unspent outputs (and only unspent outputs) may be used to effect further transactions. The requirement that only unspent outputs may be used in further transactions is necessary to prevent double spending and fraud.

For this reason, inputs on a blockchain are deleted when a transaction occurs, whilst at the same time, outputs are created in the form of UTXOs. These unspent transaction outputs may be used (by the holders of private keys; for example, persons with cryptocurrency wallets) for the purpose of future transactions.

CHAPTER 11 51% Attack

A 51% attack is a method of disrupting the consensus state of a blockchain, potentially tampering with transactions or re-ordering blocks. 51% attacks can occur when one or multiple entities control 51% of the mining hash power (in a Proof of Work network) or 51% of the token supply (in a Proof of Stake network).

Such attacks can occur through individual entities, such as a large stakeholder in a PoS network, or coordinated groups of entities, such as mining pools in a PoW network. During the time such an entity has over half of the hashpower or stake, it can reverse transactions, double spend, prevent some transactions from being included in blocks, produce all blocks, and prevent other entities from creating blocks. However, the attacker cannot enforce changes to the protocol codebase, change block rewards, or spend other's tokens. [12]

In Blockchains generally, what we really care about is transaction finality. We want to know when we receive money that it's our money, and that nobody can take that money away from us in the future. Proof of Work provides a powerful stamp that says 'this history cannot be changed without doing a lot of work'. When we get money in Bitcoin, we know that the only way we can lose that money is if an alternate history appears that doesn't include our payment, and that alternate history has more work than the history we see.

We also know that alternate histories are really, really expensive. Proof of Work provides an ingenious tether to resources in the physical world—we know that a block requires doing a ton of computation, and we know from the laws of physics that this type of computation is inherently very energy expensive. (There are types of computation that don't require much energy, or at least they require very little, whereas PoW is not among them.) We know when we see a Bitcoin block, it took tens of thousands of dollars in electricity to produce, even using the most sophisticated hardware in the world. If someone is also making an alternate history, they are required by the laws of physics to also be spending at least that much money on their alternate history.

In terms of guarantees from the laws of physics, this is more or less where it stops. A Bitcoin block costs tens of thousands of dollars to produce, and a one block double-spend necessarily costs tens of thousands of dollars. If people are waiting 6 blocks, then an attacker trying to execute a 6 block double-spend is going to need to spend over one hundred thousand dollars executing their attack.

When you think about the sheer volume of transactions on the Bitcoin network, this doesn't paint a great picture, especially when you consider that an attacker could double-spend multiple people/services/exchanges all at once. If we stop here, it more or less becomes unsafe for the aggregate transaction volume of a single block to exceed the block reward. In practical terms though, an attacker is unlikely to be able to simultaneously double-spend every single participant in a block, especially if there are hundreds or thousands of participants, and many of them have trust with each other (meaning, you trust your friend not to double spend you when they send money, or you otherwise have some recourse if the transaction does get double-spent).

The requirement for building an alternate history is more than just a requirement of spending money on electricity. You need hardware that can convert that electricity into an alternate history. And even better, we know that the original history is going to be extended by all the non-attacking miners on the network, so you need access to more hardware than the rest of the network combined—a 51% attack.

Caveat: What matters is not only the volume of hardware, but the speed and efficiency of that hardware as well. The speed at which your hardware can convert electricity into Proof of Work is called hashrate, and for any double spend attempt to succeed, you need more hashrate than the rest of the network combined. And the efficiency of your hardware determines exactly how much money on electricity you'll have to spend to produce that alternate history. If your hardware is half as efficient (most mining hardware can see significant speed boosts if they are willing to sacrifice efficiency), then a one hundred thousand dollar attack turns into a two hundred thousand dollar attack.

1. Proof of Work provides a cryptographic assurance that a certain amount of money needs to be spent to create an alternate history. Therefore, any attack (double-spend or otherwise) using alternate histories must minimally have a payoff that is larger than the cost to create that alternate history (even if you already have 100% of the hashrate).

2. The original history is being continuously extended by the network. A successful alternate history requires having more hashrate than the rest of the network combined. So the barrier

for creating alternate histories is higher than just being able to afford the electricity, you also need access to billions of dollars of hardware.

3. Specialized hardware is both faster and more energy efficient than standard hardware, by so many orders of magnitude that at least in Bitcoin and other ASIC mined coins, using ASICs is the only practical means to build an alternate history.

4. ASICs are very expensive, and inherently useful exclusively for mining that particular coin. So if the ASIC is not mining, it's just wasted money, and it's a LOT of wasted money. This makes it unlikely that a meaningful number of ASICs exist today which are not actively mining, simply because its so expensive for them to exist and then do nothing. For ASIC mined coins only, this allows us to worry less about how expensive it is to build an alternate history and instead focus more on who owns the existing hashrate—that hashrate likely represents all practical hashrate that can be applied to attack the network.

5. For ASIC mined coins only, there is a very large amount of hardware that is useful exclusively for mining a particular coin. If the value of that coin falls, the value of the hardware necessarily falls with it. Performing attacks that could shake market confidence, scare away users, or otherwise affect the price of the coin ends up being a lot more expensive for the owner of the hashrate than just the money spent on electricity—they also have to consider the losses they incurred when the value of their hardware dropped.

On the Impotence of 51% Attacks

51% attacks are a lot less powerful than most people realize, and it's one of the greatest strengths of Bitcoin. Miners are beholden to the consensus rules. If miners create an illegal block, it doesn't matter how much hash rate they have or how much they extend the illegal chain—full nodes will just ignore them. This means that miners are unable to change consensus rules like the coin inflation or block size. Miners are unable to steal money that was never sent to them, and they can't force full nodes off of the network. [13]

CHAPTER 12 Technical Implemementation

The main part of our implementation is to create a project that should give the opportunity to anybody to see how the blockchain can function.
My project is consisted of 2 sub - projects:

- Blockchain
- Cryptocurrency

The first of project is consisted mainly with the main function that a blockchain should follow in order to work properly and afterwards the “Cryptocurrency project” based on the “Blockchain Project” ,which implements the functionalities that we explained in theory before in our previous chapters.

Main logic behind the above projects :

- Create a block (Creation, genesis block , mine blocks , Sha256 Hash function)
- Create Chain (Chain Validation , Replace chain)
- Proof of work (PoW && Nonce, Dynamic Block Difficulty)
- Wallets and Transactions on the Blockchain (Creation of wallets/transactions, sign/verify transactions)
- Transaction Pool – Transactions with wallet
- Mine Transactions in a Block (Miners of transactions, reward transactions, calculate the wallet balance between each transaction, balance calculation)

<u>Blockchain Project Analysis</u>

<u>Constants Class (Description)</u>

Difficulty: There are several blocks in a given block chain. The first block is the Genesis block. Then the next block the second block at a certain block and so on. These blocks form a linked list where the nodes are cryptographically linked together. So, the previous hash over the third block is the actual hash value of the second block. So, that's why the blocks are cryptographically linked together. Also, Minors are going to generate these hash values but there are some constraints as far as these hash values are concerned. There is, a parameter called difficulty and it is characterized by leading zeros. The aim of mining is to generate hashes but there are some constraints and because of these constraints most of the generated charge 256 hashes are not a load. The difficulty of mining is defined by the leading zeros. What's extremely important that the leading zeros we have the harder it is to find a given hash. And this is why we have this known integer - nonce- within the given block.

<u>SHA256Helper Class (Description)</u>

The miners are going to increment the value of the nonce in order to generate new SHA256 hashes and generated hash with the right amount of leading zeros is going to be the final hash value of the given block.

THE DIFFICULTY OF MINING IS DEFINED BY THE LEADING ZEROS

<u>Why is it called Difficulty?</u>

Because the more leading zeros are there, the harder to find that given hash.

One leading “0”: P (finding hash with 1 leading zero) = **hashes with 1 leading zero** / **total number of hashes** = $\mathbf{16^{63}/16^{64}} = 6{,}25\%$

Two leading “0”: P (finding hash with 2 leading zero) = **hashes with** 2 **leading zero** / **total number of hashes** = $\mathbf{16^{62}/16^{64}} = 1/256 = 0.39\%$

Eighteen leading “0”: P (finding hash with 18 leading zero) = **hashes with** 18 **leading zero** / **total number of hashes** = $\mathbf{16^{48}/16^{64}} = 2.1\times10^{-20}\%$ (this is the actual difficulty of bitcoin)

<u>SHA256Helper Class (Description)</u>

This is a static method that's going to get an input. What's going to be the input the block itself and it is going to return the hexadecimal Sha256 representation of the given block. We are able to get the hash as a one dimensional array of bytes :

```
byte[] hash = digest.digest(data.getBytes("UTF-8"))
```

but of course we would like to end up with a hexadecimal representation.

And we are going to transform this byte into a 64 character long hexadecimal string. And finally, we just have to call the toString method on the string buffer in order to end up with the short 256 value. This is how we will generate the hash values.

Block Class (Description)

1. public Block(int id, String transaction, String previousHash)
2. public void generateHash()
3. public String getHash()
4. public String getPreviousHash()
5. public void incrementNonce()
6. public String toString()

The block is going to be the fundamental building block of the block chain itself.
It has several of their valuables such as the id of the block such as the time stamp the hash of the block , the previous hash in the block chain. We have the transactions and we have a nonce parameter.
The first block is called the Genesis block and we have to first block the second block disserved block and so on. These blocks are cryptographically linked together based on the hash values.
So that's why the previous hash value of the third block is the same as the hash value in the previous block.
We just have to define the getters and setters as far as the hash value and the previous hash values are concerned. We have given a method increment nonce parameter.
So when given a minor get a given block and it tries to find the hash value it is going to increment the value of the nonce until the given constraint is Matt was constrained.
This is why we have defined the difficulty to be five because the constraint is that the 64 character long hexadecimal hash has to start with five zeros. Within the block the minor is going to increment the nonce parameter in order to generate new hash values.
So basically this is why we have these generate hash function.
It is going to get the values within the block as far as Id, nonce, timestamp , ,previous hash and the transaction is concerned and it is going to call these short 256 helper generate hash functioning order to generate the hash value of that given block, this hash value is going to identify the block. So it is something like a fingerprint .
So the block has several variables, Id, nonce, timestamp previous hash , transaction and we are going to use these values in order to generate the fingerprint so the hash of the given

block and as far as mining is concerned ,the miner is going to increment these nonce in order to generate new hashes for the given block.

Blockchain Class (Description)

1. public BlockChain()
2. public void addBlock(Block block)
3. public List<Block> getBlockChain()
4. public int size()

We are going to store the blocks in a array list. We instantiate a list of blocks. It's going to be the block chain and we instantiate this latest as a new array list because the blockchain itself is just the sum of the single blocks. Genesis block , block one, block two ,block three ,these are going to form the blockchain and they are cryptographically linked together.

We have some getters and setters. We have the addblock method , and this add block is going to have a block as an argument and it is going to add the given block to the block chain.

We have a size method that's going to return the size of the block chain. And we have a to string method that's going to iterate through the blocks within the block chain and it's going to printout these single blocks.

Miner Class (Description)

Every miner is going to get a reward because they are validating the given transactions. Debt in a centralized system we have trusted third parties such as banks and the banks are going to verify the given transactions as far as de-centralized Ledger is concerned. So somehow we have to make sure that these transactions will be validated and this is why mining came to be miners will handle and verify the transactions they are getting paid.
But this is not the aim of mining. It is just the by-product miners who will handle and verify the transactions. For this reason first of all they will get a reward because of the mining procedure and this is why we have defined.
Mining means finding the right hash value for the given block. But there are some constraints as far as difficulty is concerned.

What is it mean the difficulty is five? It means that there has to be five leading zeros at the beginning of every hash. So, miners will generate hash values until they find the right hash.

What does it mean?

Basically the golden hash is the right hash value. So, the hash value with the right amount of leading zeros. That's why we generate a given string leading zeros with as many zeros as we have defined in the Constants method.

Because the difficulty is five, that's why we will have a string The five leading zeros.

What does it mean generate Hashes?

We have already implemented this method that the given miner will take these values and they are going to use the SHA256 Helper in order to generate the hash value and it hash the value of the given block to be generate the hash value. Then the algorithm is going to check whether they're given hash is valid or not. (notGoldenhash).

What does it mean that the hash is valid?

That it contains the right amount of leading zeros. And basically, if the generated hash is not good then the minor will increment the nonce in the block. But anyways it is going to increment the nonce variable.

Why is it good?

Because if we have another value for the nonce and generate again the hash value because this hash value depends on the nonce. That's why the generated hash will be different again. So if we tamper the nonce then the generateHash method is going to return a different hash value and basically mining is that simple:

```
while(notGoldenHash(block)) {
        block.generateHash();
        block.incrementNonce();
    }
```

The miners real looking for the golden hash with the right amount of leading zeros. And if the miner manages to find that given hash first of four, we printout that the given block has just mined right the given hash value and the minor will order that the given block to be added in the block chain. And because it's going to take some time for the miner to find this hash value and there are some costs associated with this operation because it is extremely expensive to generate this huge amount of hashes. That's why the miner people get a reward in this case the miner will get 10 bitcoins.

Thus, collectively , miner class is going to have two methods.

There are the **notGoldenHash boolean method** that's going to check whether the given generated hash value is valid or not. And it has a viral loop because the miner keeps generating new Hashes until it finds the right hash with the right amount of leading zeros. Of course, the higher the difficulty the more expensive it is to find the golden hash. And it is going to take more time to find a given hash. But anyways the miners will get a reward for this operation. This is why minors exist.

CRYPTOCURRENCY PROJECT ANALYSIS

Cryptocurrency contains below classes:

1. CryptographyHelper
2. Miner
3. Transaction
4. TransactionInput
5. TransactionOutput
6. Wallet

CryptographyHelper Class:

generateHash

public static KeyPair ellipticCurveCrypto :
This method is used to generate a keypair object. Ensure the encryption of the transactions and also that funds can only be spent by their rightful owners. For this reason, we use Elliptic Curve Digital Signature algorithm with a private and public key. The private key is a secret number 256 bits long integer known only to the person that generated it. And then we are able to generate a public key based on this private key which is basically a coordinate on an elliptic curve.

public static boolean verifyECDSASignature(PublicKey publicKey, String data, byte[] signature)

Miner Class

1. public void mine(Block block, BlockChain blockChain)
2. public boolean notGoldenHash(Block block)
3. public double getReward()

*Very similar with Blockchain implementation

Transaction Class

1. // Constructor:
// public Transaction(PublicKey sender, PublicKey receiver, double amount, List<TransactionInput> inputs)

2. public boolean verifyTransaction()
3. public double getInputsSum()
4. public void generateSignature(PrivateKey privateKey)
5. public boolean verifySignature()
6. private void calulateHash()
7. public String getTransactionId()
8. public void setTransactionId(String transactionId)
9. public PublicKey getSender()
10. public PublicKey getReceiver()
11. public double getAmount()
12. public byte[] getSignature()
13. public List<TransactionInput> getInputs()
14. public List<TransactionOutput> getOutputs()

<u>TransactionInput Class</u>

1. public TransactionInput(String transactionOutputId)
2. public String getTransactionOutputId()
3. public TransactionOutput getUTXO()

<u>TransactionOutput Class</u>

1. public TransactionOutput(PublicKey receiver, double amount, String parentTransactionId)
2. private void generateId()
3. public boolean isMine(PublicKey publicKey)
4. public String getId()
5. public StringgetParentTransactionId()
6. public PublicKey getReceiver()
7. public void setReceiver(PublicKey receiver)
8. public double getAmount()
9. public void setAmount(double amount)

<u>Wallet Class</u>

1. public Wallet()
2. public double calculateBalance()
3. public Transaction transferMoney(PublicKey receiver, double amount)
4. public PublicKey getPublicKey()
5. public PrivateKey getPrivateKey()

Every single user in the network has a wallet and that’s why every single wallet has a private key. There’s a public key associated with every single wallet because this public key

is generated from this private key. So, we can use this cryptography how elliptic curve crypto method.
This method is going to generate a key pair. That's a container for the public key and private keys. We will use the elliptic curve cryptography in order to generate these key pairs.
And we can get the private key and we can get the public key. That’s why every single wallet has a private key and public key. The public key is open to everyone in the networks and every single user can know the public key. This is how they can verify your given transaction signed by the private key.

We have two important methods in order to calculate the balance.

<u>What do we have to do?</u>
We just have to consider the unspent transaction outputs (UTXO’s), the blockchain that UTXO’s has. Blockchain has a map with extreme keys and transaction output values. It is the UTXO’s. This is the data structure that stores all the and transactions.
So in this case these transactions are present in the UTXO’s data structure and if we want to calculate the balance of a given wallet we just have to sum up the beat going of values associated with the given user.We have the balance initialized to be zero where we can iterate through all the UTXO’s.

<u>What does it mean the UTXO’s stores transactions?</u>
So we consider all the transactions present in the <u>UTXO’s</u>, those data structure we get the transaction output.
So because this blockchain has the <u>UTXO’s</u> with transaction output values that's why we get transaction output and if the transaction output belongs to us so is the mine we can compare transaction outputs based on public keys.

The miner method in the transaction output is going to get a public key. And if this public key is equal to the receiver or public it means that this transaction output would belong to the given user. <u>And if this transaction belongs to this public key what does it mean?</u> That means that this Wallet belong to this user in the network. After this, the balance is incremented by the transaction output amount. This is the way on how we calculate the total balance of the given Wallet.

<u>How can we transfer money?</u>
We have to define a given receiver with a public key. Every single user and participants of the network has a public key. This is how we can identify them.

So , we have a receiver and we have a given amount. We would like to transfer and first of all , if the balance is smaller than the amount we would like to see and of course it is an invalid transaction because of not enough money present in the wallet. And anyways we just have to store the inputs for the transaction in this array.
So , that’s why this input , is going to be the list of transaction inputs. We have to find our spend transactions and the block chain stores all the <u>UTXO’s</u>. So, this blockchain so is going to store all the transactions and we have to consider the transactions inputs that belongs to the given Wallet.
That's why we iterate through all the transaction outputs , where as a result we can get the value. This is a transaction output.

The transaction output , as a result , is mined, which means that these UTXO's belongs to this Wallet ,with these private key, then we are going to own that given transaction import to the imports list and then we just have to create a new transaction with the inputs with a public key with the receiver and the given amount. What does it remain to do? We just have to do the generation the signature and after this, we just have to return this new transaction.

Now it's clear , that every single transaction has a sender, receiver, an amount and a given user with the given wallet is going to use his or her private key in order to generate the signature and any other member of the network can verify that a given transaction with the given signature ,if it belongs to the owner.

We have to create a new transaction and then we have to generate the signature with the private key. With the private key is going to send a given fund to the receiver with that amount of money. That's why, firstly, we have to find the unspent transactions because these unspent transactions define that.

Outputs of above implementation

The outputs of the above technical implementation are better appeared in my personal repository on this link : Thesis Repository [14] . On my Thesis Repository, you could find the source code of my implementation.

Conclusions

During the implementation of these two projects we managed to see how we can create a block and a blockchain by creating firstly, the Genesis block and then the other and after all how to mine each block. Additionally, by using specific algorithms, we created a wallet and manage to sign and verify each transaction. We managed to take advantage from the transaction pool, by transacting effectively with the wallet. Also, we showed how we can calculate the wallet balance after an exchange, and which is the process in order to give the reward to the miners after a successful transaction (and the opposite).

In Blockchain Project, we construct each block where the main scope is to add each block in the blockchain. Firstly, we construct the Genesis block. Each block has some specific values, which if they are not exist, the block cannot be mined, and cannot be added to any blockchain. Which are they? The ID of the transaction, the number of the transaction, the hash of the block, the previous hash of the block. The number of the transaction, apparently means, the nonce value. Any change to the block data (such as the nonce) will make the block hash completely different. Which makes our functionality really aligned with the theoretical approach.

In Cryptocurrency project, we construct the genesis block and we calculate the balance of the User A. After User A tries to send money to User B, where our implementation checks if the transaction is valid. If it is valid, then the transaction is added to the block, and we calculate the new balance for both Users. If transaction is not valid, then we send back a message of rejection of this transaction. At the end of all of these transactions, we print the hash value of the transaction and the wallet balance of its User.

CHAPTER 13 Blockchain Use Cases

In this Chapter our main scope is to present some aspects of Blockchain and it can affect the world in the near future.

By removing the need for trust and expensive security, blockchain provides improved efficiency. Moreover, the decentralized network can be configured as a transparent database, visible by all participants. In this sense, blockchain technology provides the ability to create a distributed yet unified record. This offers opportunities for improving performance and security in many industries and organizations (e.g., charity, supply chain, healthcare, etc.).

Charity

Many charitable organizations around the world strive to deal with challenges of resource management, operational transparency, and effective governance. Blockchain technology can certainly help these foundations optimize the process of receiving and managing funds.

We already have some notable examples of the integration of blockchain technology into charity. For instance, the Blockchain Charity Foundation (BCF) is a non-profit organization that works toward sustainable development goals to fight poverty and inequality, aiming to enable blockchain-powered philanthropy around the world.

Supply chain

Most supply chain networks are facing many obstacles in regards to transparency and efficiency. The current management system still depends on trust and is far from providing proper integration between the companies and parties involved. Blockchain technology can be used to track the whole process of creating and distributing materials within a supply chain network. A distributed database may suit well for securely recording any related data, ensuring the authenticity of the products, as well as the transparency of payments and transportation. imagine that a supplier is transferring the food products to an importer who verifies that the supplier, country, and food type all match the correct identifiers. At the port of entry, the supplier is again checked against a list of known suppliers in a database (managed by the regulator). If the supplier is of type exempt, then the products are transferred to the retailer. If the supplier is non-exempt, the products are checked against a list of known food products in a database (managed by the regulator). If the food is an exempt product, then it is transferred to the retailer. If the food is non-exempt, the importer must conduct the hazard analysis (either independently or by using a third party). The supplier provides the hazard analysis report to the regulator. The regulator reviews compliance and transfers the products to the retailer. This pattern captures the regulatory compliance logic for the FDA Foreign Supplier Verification Program in a smart contract that's deployed on a business network.

Healthcare

Operational bottlenecks, data errors, and bureaucracy are a significant concern for the healthcare industry. Blockchain has several use cases in healthcare, including tracking drugs through the supply chain and managing patient data.
Moreover, blockchain may offer significant security benefits to hospitals, since these institutions are often attacked by hackers due to the high value of data they hold and their high dependability on it.

Companies are exploring the use of blockchain as a way to store digital health records. Such solutions can reduce overall expenses while also enhancing data privacy and accuracy.

Royalty payments

Musicians, video game creators, and artists in general often struggle to get the pay they deserve due to digital piracy, unfair relationship with third-party agencies, or simply by not being paid royalties that are due.

Blockchain technology can be used to create a platform where creative talents have an immutable and transparent record of who is renting, buying, and/or using their content. Such a platform can also facilitate payments through smart contracts - which are basically self-executing digital contracts.

Governance

Blockchain technology has the potential to greatly improve governance in various different sectors. By managing networks and operations in a more democratized, fair and secure manner, blockchain-based systems may be implemented as a tool to eliminate vote fraud and increase trust during elections or other constitutional processes. They may also be utilized as a powerful weapon against corruption, enhancing data integrity and traceability in a variety of scenarios, from tax collection to financial aid distributions.

Payment solutions and DApps

When it comes to sending money worldwide, blockchain technology has already proven to be very efficient. Sending cryptocurrencies to friends, families, and others around the world is already cheaper and faster compared to what centralized banks and payment solutions have to offer.

Moreover, centralized websites and Apps don't give users control over their data and often don't reward them according to the true value they bring to the platform. Blockchain-based decentralized applications (dApps) take out the middleman, giving users the potential to enjoy reduced fees, better incentives, and greater transaction efficiency, while also being able to send and receive digital money.

As Vitalik Buterin once said, blockchain solutions allow people to work directly with one another, removing the need for intermediaries or centralized systems.
“Whereas most technologies tend to automate workers on the periphery doing menial tasks, blockchains automate away the center. Instead of putting the taxi driver out of a job, blockchain puts Uber out of a job and lets the taxi drivers work with the customer directly.”

Internet of Things (IoT)

Blockchain and the Internet of Things (IoT) are a natural match. Blockchain is decentralized technology and IoT networks are often used to collect data from sources that are scattered apart.
Blockchain allows organizations to keep an immutable and transparent ledger of IoT devices, the data they collect, and the interactions between one another. Among its security features and its cryptocurrency applications, blockchain offers an ideal platform for machine-to-machine (M2M) transactions.

As blockchain is a technology that's based on facilitating accurate and secure transactions, it only makes sense that it's integrated with IoT to ensure accountability and data accuracy and security. That's why many firms have been placing a lot of resources into a blockchain-powered IoT network.

Positives and Negatives of Blockchain

Advantages

Distributed

Since blockchain data is often stored in thousands of devices on a distributed network of nodes, the system and the data are highly resistant to technical failures and malicious attacks. Each network node is able to replicate and store a copy of the database and, because of this, there is no single point of failure: a single node going offline does not affect the availability or security of the network.

In contrast, many conventional databases rely on a single or a few servers and are more vulnerable to technical failures and cyber attacks.

Stability

Confirmed blocks are very unlikely to be reversed, meaning that once data has been registered into the blockchain, it is extremely difficult to remove or change it. This makes blockchain a great technology for storing financial records or any other data where an audit trail is required because every change is tracked and permanently recorded on a distributed and public ledger.

For example, a business could use blockchain technology to prevent fraudulent behavior from its employees. In this scenario, the blockchain could provide a secure and stable record of all financial transactions that take place within the company. This would make it much harder for an employee to hide suspicious transactions.

Trustless system

In most traditional payment systems, transactions are not only dependent on the two parties involved, but also on an intermediary - such as a bank, credit card company, or payment provider. When using blockchain technology, this is no longer necessary because the distributed network of nodes verify the transactions through a process known as mining. For this reason, Blockchain is often referred to as a 'trustless' system.

Therefore, a blockchain system negates the risk of trusting a single organization and also reduces the overall costs and transactions fees by cutting out intermediaries and third parties.

Disadvantages

51% Attacks

The Proof of Work consensus algorithm that protects the Bitcoin blockchain has proven to be very efficient over the years. However, there are a few potential attacks that can be performed against blockchain networks and 51% attacks are among the most discussed. Such an attack may happen if one entity manages to control more than 50% of the network hashing power, which would eventually allow them to disrupt the network by intentionally excluding or modifying the ordering of transactions.

Despite being theoretically possible, there was never a successful 51% attack on the Bitcoin blockchain. As the network grows larger the security increases and it is quite unlikely that miners will invest large amounts of money and resources to attack Bitcoin as they are better rewarded for acting honestly. Other than that, a successful 51% attack would only be able to modify the most recent transactions for a short period of time because blocks are linked through cryptographic proofs (changing older blocks would require intangible levels of computing power). Also, the Bitcoin blockchain is very resilient and would quickly adapt as a response to an attack.

Data modification

Another downside of blockchain systems is that once data has been added to the blockchain it is very difficult to modify it. While stability is one of blockchain's advantages, it is not always good. Changing blockchain data or code is usually very demanding and often requires a hard fork, where one chain is abandoned, and a new one is taken up.

Private keys

Blockchain uses public-key (or asymmetric) cryptography to give users ownership over their cryptocurrency units (or any other blockchain data). Each blockchain account (or address) has two corresponding keys: a public key (which can be shared) and a private key (which should be kept secret). Users need their private key to access their funds, meaning that they act as their own bank. If a user loses their private key, the mone y is effectively lost, and there is nothing they can do about it.

Inefficient

Blockchains, especially those using Proof of Work, are highly inefficient. Since mining is highly competitive and there is just one winner every ten minutes, the work of every other miner is wasted. As miners are continually trying to increase their computational power, so they have a greater chance of finding a valid block hash, the resources used by the Bitcoin network has increased significantly in the last few years, and it currently consumes more energy than many countries, such as Denmark, Ireland, and Nigeria.

Storage

Blockchain ledgers can grow very large over time. The Bitcoin blockchain currently requires around 200 GB of storage. The current growth in blockchain size appears to be outstripping the growth in hard drives and the network risks losing nodes if the ledger becomes too large for individuals to download and store. [15]

Βιβλιογραφία

[1] D. M. v. Rijmenam, «What is the Blockchain and Why is it So Important?,» 2016. [Ηλεκτρονικό].

Available: https://www.linkedin.com/pulse/what-blockchain-why-so-important-mark-van-rijmenam/.

[2] W. P. &. T. v. Werkhoven, «On the Secure Hash Algorithm family (Chapter 1 of Cryptography in Context),» 2008.

[3] V. Buterin, «The Meaning of Decentralization,» [Ηλεκτρονικό]. Available: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[4] Lisk Academy, «Consensus Protocols,» 2019. [Ηλεκτρονικό]. Available: https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/consensus-protocols.

[5] R. S. &. M. P. Leslie Lamport, «The Byzantine Generals Problem,» 1982. [Ηλεκτρονικό]. Available: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf.

[6] M. Deneuville, «An in-depth guide into how the mempool works,» 2016. [Ηλεκτρονικό]. Available: https://blog.kaiko.com/an-in-depth-guide-into-how-the-mempool-works-c758b781c608.

[7] H. Knutson, «What is the math behind elliptic curve cryptography?,» 2018. [Ηλεκτρονικό]. Available: https://hackernoon.com/what-is-the-math-behind-elliptic-curve-cryptography-f61b25253da3.

[8] C. Research, «SEC 2: Recommended Elliptic Curve Domain Parameters - Secp256k1,» 2010. [Ηλεκτρονικό]. Available: http://www.secg.org/sec2-v2.pdf.

[9] B. K. Kikwai, «Elliptic Curve Digital Signatures and Their Application in the BitcoinCrypto-currency Transactions,» 2017. [Ηλεκτρονικό]. Available: http://www.ijsrp.org/research-paper-1117/ijsrp-p7117.pdf.

[10] «Unspent Transaction Outputs (UTXO),» [Ηλεκτρονικό]. Available: https://sci.smithandcrown.com/glossary/unspent-transaction-outputs-utxo.

[11] N. Satoshi, «Bitcoin: a peer-to-peer electronic cash system,» 2008. [Ηλεκτρονικό].

[12] «51% Attack,» [Ηλεκτρονικό]. Available: https://sci.smithandcrown.com/glossary/51-attack.

[13] D. Vorick, «Choosing ASICs for Sia,» 2017. [Ηλεκτρονικό]. Available: https://blog.sia.tech/choosing-asics-for-sia-b318505b5b51.

[14] K. A. Dionysios, «Thesis Technical Repository,» 2019. [Ηλεκτρονικό]. Available: https://github.com/adacapo21/Thesis/.

[15] Binance Academy, «Blockchain Advantages and Disadvantages,» 2019. [Ηλεκτρονικό]. Available: https://www.binance.vision/blockchain/positives-and-negatives-of-blockchain.

[16] https://www.monetha.io, «Understanding the Merkle Tree and How It Works,» [Ηλεκτρονικό]. Available: https://medium.com/@monetha/understanding-the-merkle-tree-and-how-it-works-cc7a84ae6a92.

[17] S. H. &. W. S. Stornetta, «How to Time Stamp a Digital Document,» 1991.

[18] S. D. &. J. P. Chris Berg, «The Blockchain Economy: A beginner's guide to institutional cryptoeconomics,» 2017. [Ηλεκτρονικό]. Available: https://medium.com/@cryptoeconomics/the-blockchain-economy-a-beginners-guide-to-institutional-cryptoeconomics-64bf2f2beec4.

[19] A. Corbellini, «Elliptic Curve Cryptography: a gentle introduction,» 2015. [Ηλεκτρονικό]. Available: https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/.

[20] M. Deneuville, «How do Mempools Work?,» 2016. [Ηλεκτρονικό]. Available: https://blog.kaiko.com/an-in-depth-guide-into-how-the-mempool-works-c758b781c608.

[21] N. Acheson, «How Bitcoin Mining Works,» 2018. [Ηλεκτρονικό]. Available: https://www.coindesk.com/information/how-bitcoin-mining-works/.

[22] G. Konstantopoulos, «Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance,» 2017. [Ηλεκτρονικό]. Available: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419.

[23] A. Castor, «A (Short) Guide to Blockchain Consensus Protocols,» 2017. [Ηλεκτρονικό]. Available: http://www.coindesk.com/short-guide-blockchain-consensus-protocols.

[24] L. e. a. Lamport, «The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4(3): 382-401,» 1982. [Ηλεκτρονικό]. Available: https://dl.acm.org/citation.cfm?id=357176.

[25] M. L. B. Castro, «Practical Byzantine fault tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation,» 1999. [Ηλεκτρονικό]. Available: http://pmg.csail.mit.edu/papers/osdi99.pdf.

[26] M. J. A. Jakobsson, «Proofs of work and bread pudding protocols,» 1999. [Ηλεκτρονικό]. Available: http://www.hashcash.org/papers/bread-pudding.pdf.

[27] D. Tapscott, «How the blockchain is changing money and business,» [Ηλεκτρονικό]. Available: https://www.ted.com/talks/don_tapscott_how_the_blockchain_is_changing_money_and_business/transcript?language=en.

Appendix A

Technical Implementation : https://github.com/adacapo21/Thesis

Curriculum Vitae

Experience

ALTEN SA (mission AMADEUS IT Group) June 2018 to Current

Software Engineer

Sophia Antipoles , France, Provence-Alpes-Côte d'Azur

- C++ , Regression Tests - Unit Tests , Python , OracleDB , Mercurial/ Git
- Analyse user requirements and develop software according to Amadeus standards.
- Coordinate project delivery with Product Definition, QA and other DEV teams.
- Interface with relevant divisions and departments to identify interactions with other Amadeus applications and ensure technical compatibility
- Conduct unit, package and performance tests of the software and ensure a level of quality in line with the Amadeus guidelines.
- Perform code reviews in line with Amadeus quality standards.
- Participate in the validation/acceptance phase of the product cycle ensuring the fine-tuning necessary to finalize the product. Participate to the sizing of change requests.
- Produce software documentation.
- Support the production phase by debugging existing software solutions in response to Problem Tracking Records (PTR) and Incident Records (IR).
- Build and maintain industry knowledge.

Synectics January 2018 to April 2018

Developer

Athens

- Implements integration through interfaces with third party systems (CRM, ERP, Web portals)
- ASP.NET MVC, Entity Framework , Microsoft SQL Server , Visual Studio

- Team Foundation Server , Windows Presentation Foundation
- Language Integrated Query (LINQ) , Javascript , jQuery
- Produced efficient codes based on specifications.
- Effectively handled the communications between in-house software team, clients, and stakeholders.

Hellenic Army January 2017 to September 2017

Software Engineer

Athens

- Heavily involved in the EU project TOXI-TRIAGE (H2020). Development and implementation of Simulation Software of detection of CBRN incidences. (http://toxi-triage.eu/)
- Designing and developing a secondary software in JAVA, providing a graphical user interface, that was responsible for converting "JANUS" data files in Csv format, to be used in post-exercise analysis.
- Development of chemical warfare feature in the Middleware software, in order to be utilised in future CPX-CAX.
- Compliance with the security standards, as defined by the ISO 27001:2013, requiring the connection through authentication, for every remote application.

Tekmon Information Systems December 2015 to December 2016

Front End Developer

Athens

- Assisted in Front-end Engineering
- JavaScript , Frameworks: AngularJS, NodeJs, Bootstrap
- Created successful websites that met requirements for objectives such as load speed and design.
- Developed web-site mock-ups for clients to ensure quality control and client satisfaction before project development phase.

Education and Training

National and Kapodistrian University of Athens 2017

Master of Science: **Control and Computing**

Athens

National and Kapodistrian University of Athens 2014

Bachelor of Science: **Physics**

Athens

Certifications

- **Blockchain**, a 4-course specialization by The State University of New York & University at Buffalo on Coursera. Specialization Certificate earned on March 2, 2019
- "**Research - Software Engineer**" Certification from Centre of Information Technology Support of the Hellenic Army (KE.P.Y.ES.)
- Course Certificate on Front-End JavaScript Frameworks: AngularJS: The Hong Kong University of Science and Technology