Contents

Chapter 1

Evaluation and Comparison of Blockchain Consensus Algorithms

Joel Barmettler, Özgür Acar Güler, Marc Laville, Spasen Trendafilov

Abstract *- While the public catched the interest in blockchains through cryptocurrencies like Bitcoin, the technology behind these currencies emerged to a general field of research around secure, decentralized and trustless computing system. Fundamentally, these systems are formed by nodes that all have a redundant copy of a shared data structure, in most cases an immutable blockchain. In order to update this data structure among all nodes consistently, the nodes run a consensus protocol that ensures save and consistent modification of the distributed ledger among all nodes participating in the network. Over the years, a number of different consensus algorithms formed. This work discusses a few of the most influencal, controversial or innovative consensus algorithms and established a comparison based on a selection of key features.*

Contents

1.1 Introduction

With the success of Bitcoin over the past years, the Blockchain technology has rapidly gained popularity among computer scientists as well as in the financial sector. Although the hype surrounding the cryptocurrencies has slowed down rapidly as the currencies lost in value, the technology behind them remains a fast growing scientific field.

In [61], a concept similar to blockchain was first described. The goal of [61] was to come up with a way to digitally time-stamp a document in a way that can not be manipulated and that does not require a service to keep track of the records [61].

In 2008, someone known under the pseudonym of "Satoshi Nakamoto" came up with the first concept of a blockchain as it is known today. In 2009, the concept was implemented and builds the core technology behind the cryptocurrency Bitcoin to this day.

A blockchain is a data structure that stores facts like transactions in an time-ordered manner without a central party. The fundamental concept of a blockchain and most Distributed Ledgers, a broader term for decentralized data structures, is creating a secure environment in which participants don't have to trust each other. While in classical computing systems, data is stored in large databases that are in complete control of whoever owns the dedicated server infrastructure, therefore relying on trust that nobody alters the data without the users knowledge, distributed ledgers eliminate all central authorities by decentralizing both the data and the data-manipulation algorithm. Data in a distributed ledger is stored redundantly throughout computers across the world, ensuring that no single person can alter the content of the ledger, making it a secure medium to permanently store data.

By using a decentralized system in which all participants store their own version of the blockchain, a blockchain is invulnerable to some risks a centralized system is vulnerable to. It is for example harder to delete or modify data if every node in the system stores a copy of the data, because to do so, every node would have to be attacked. In comparison with a centralized system, only a single point of failure (SPOF) would have to be successfully attacked. The same principle of risk reduction can be applied for technical failures such as hardware failures: if a single node fails, the data is still secured on all other nodes who carry a copy of the data.

Most ledgers are publicly visible and anyone can find and locate any facts stored in the data blocks. To add new data to the ledger, the computers that participate in the network have to validate whether the request is legit or not. In the case of Bitcoin, these computers, also called network-nodes, check for each new transaction request whether the person trying to spend money also has sufficient funds. If the nodes come to a consensus that the transaction is indeed valid, they all alter their local ledger state.

A distributed ledger can therefore be characterized by what data structure the ledger has and what consensus protocol the nodes are running. While the data structure of most distributed ledgers is the famous blockchain, the consensus algorithms are the reason why blockchains are challenging to engineer. Not only does the network need to find consensus among nodes that are distributed around the whole world, it also needs to deal with latency, corrupted data, network attacks and malicious nodes.

There are three different fields types of blockchains [18]: Permissionless blockchains, permissioned blockchains and private blockchains. While permissionless blockchains are open for anybody to invoke transactions on the blockchain, run a validating node or participate in the consensus finding mechanism, permisssioned blockchains only allow limited and validated entities to run the network: Nodes are identified and controlled, transactions request often limited to known parties. Private blockchains are a subsection of permissioned blockchains with only one organization operating all parts of the blockchain, creating a single trust domain. Permissionless blockchains like Bitcoin, Ethereum or Litecoin represent the understanding of a traditional blockchain, while permissioned blockchains like Ripple or private blockchains like Hyperledger have their own legitimate applications. The type of a blockchain defines which consensus algorithms may be suitable. Often, in private blockchains, there is a central list of a small number of nodes participating in the consensus protocol. The consensus algorithms can therefore assume that no malicious nodes can be spawned in high numbers, that there is low latency in the network, and that only a relatively small number of mostly trusted nodes are part of the consensus finding protocol. This looks different for permissionless blockchains: Everybody can participate in the network, implying that malicious nodes can be spawned in a large number, overflooding the network with carefully chosen information to hinder nodes from reaching consensus.

According to the Scalability-Trilemma [69], a general trade-off between decentralization, scalability and security exists. A well-scalable protocol can handle thousands of transaction requests per second. If it can resist against all known attacks as long as a fixed majority of nodes in the network are honest, it is seen as secure. Furthermore, if the network is backed by hundreds of nodes, distributed all over the world and potentially hosted by unknown parties, it can be called decentralized.

Each consensus algorithm known today seems to perfectly solve two of them while making a trade-off on the third. Thus, no consensus algorithm is superior: The trade-off leaves room for many different algorithm implementations, addressing different needs. While private blockchains are often highly centralized, they scale well and have a high security standard. Other consensus algorithms like Bitcoin prioritize decentralization and security, while failing on scalability.

1.2 Consensus

The rate of involved trust varies between blockchain types, but they are all distributed computing systems, sharing the need for a robust and stable ecosystem. In order to work properly, all nodes of the distributed system have to eventually reach consensus, even in the presence of failed or malicious nodes. There are two main tasks involved in finding consensus among distributed nodes that communicate peer-to-peer [28]. First, each node has to run a (deterministic) state machine (DSM) implementing the service the blockchain provides. Second, the nodes share their state using a consensus protocol such that each nodes performs the same actions on its state machine - they have to find consensus about what operation to apply on the current state, such that the overall network remains stable.

The first step, the DSM, is an implementation of the famous Automata as explained in [55]: Each node has a state, and new transactions can change their state. In a functioning blockchain, all synchronized nodes are in the same state. Since the State Machine is deterministic, the same transaction will result in the same, new state for all nodes. Therefore, if all nodes were synchronized in the beginning, and with applying the same transaction to all nodes, the blockchain shall never be in an inconsistent state. While it is easy to have a common state among all nodes, the hard step is having a protocol that ensures that all nodes will always change their state in the same manner. This leads to the second step, the consensus protocol. It needs to follow a few properties: All nodes, or state replicas, need to agree on the same value in a non-trivial case, meaning they do not just always agree on a trivial state change, but rather on an output at least one of the nodes vote for.

Finding consensus in a distributed system with completely reliable actors is a trivial task. However, in real world scenarios there are plenty of possible faults, from loosing connection to the other nodes over process crashes to sending duplicate messages. A reliable consensus protocol shall therefore overcome such faults, as long as the number of malicious nodes is not overwhelming.

Speaking of a distributed computer system in general, a fault-tolerant system is a system in which the overall reliability exceeds the reliability of the individual parts [33]. A traditional fault in a distributed computing environment is a computer, or node, sending wrong or missing data, which can be overcome by having a certain amount of data redundancy, allowing the system to detect errors and continue working in the absence of certain data. A fault in the system may or may not cause a failure of the system, depending on whether the fault can be properly handled. The rate to which a system is tolerant to faults without causing failure is called fault-tolerance.

Distributed systems are always dependable. The formal definition of dependability is given in [40]: ”Dependability is quality of the delivered service such that reliance can justifiably placed in this service”. To increase the dependability of a system, one either avoids fault on a node level by increasing the node quality, or increase the fault-tolerance of the overall system. For permissionless blockchain networks, absolute fault avoidance is impossible: Even if all participating nodes are in perfect condition, the network still has to deal with malicious nodes and overall churn. Blockchains therefore focus on having highly robust consensus mechanisms that ensure that even with a relatively high amount of fault in the system, the network remains up and is running correctly.

For traditional distributed computing systems, such as cloud computing systems in which many computers work together to solve one common task, consensus algorithms like Paxos and Viewstamped Replication are used that allow a certain number of nodes to fail while still reaching consensus [33]. While Paxos is well suited for controlled environments, permissionless public blockchains can not function on such consensus protocols since nodes may intentionally become malicious and work against the common goal of reaching agreement. The traditional distributed-computing algorithms need to be expanded to become fault-tolerant against any arbitrary or even malicious behaviour.

1.2.1 Different Consensus protocols

Consensus algorithms in blockchain ecosystems can have different natures. Two extremes are characterized on a spectrum in terms of the basic principle which the protocols are based on. On one end of the spectrum there are protocols which are based only on computation to prove their validity. These protocols use a computational task to determine which node to entrust with the choice of the next operation on the blockchain. A node can qualify by solving a computational task faster than its rivaling nodes. Once a solution is found, the node can propose its solution to the network, where it will be validated. Bitcoins Proof of Work (PoW) consensus algorithm is a perfect example for a purely computation-based consensus algorithm. On the other extreme of the spectrum are consensus algorithms which are based only on communication. In communication-based algorithms each node gets to vote to decide which node will be allowed to perform an operation on the blockchain. Multiple rounds of voting may take place to reach consensus. For this system to work, the nodes have to assume that the other nodes who participate are allowed to vote.

There are many different consensus algorithms which combine or modify the concept of a purely competitive based with the concept of a purely communication-based protocol to create hybrid protocols with the goal to improve the performance and eliminate their respective drawbacks.

Over the next few chapters, a closer look will be taken at different consensus algorithms by analyzing how they work. The goal is to be able to compare the found characteristics of the different algorithms and put them in relation to one another. Among the wide variety of algorithms, a selection has been made to feature the most prominently used ones. In addition to that, a set of algorithms which take a different approach will be featured as well.

1.3 Byzantine Agreement

The blockchain problem is "to allow an arbitrary large network of processors to agree on the blockchain state under the assumption that the computation power of malicious processors is bounded" [46]. Assume that the network consists of a total of n processors, or nodes, out of which f are malicious and do not behave as expected. A blockchain only works if there is a protocol that ensures all honest nodes agree on the correct state of the blockchain when a new block of transactions is created as long as f is smaller than n by a known degree.

The literature describes a problem of reliable computer systems using a metaphor with Byzantine generals [44]. A group of Byzantine generals, each controlling a part of the Byzantine army, camp around an enemy city. The forthcoming battle may only be successful if all loyal generals attack at the same time, *e.g.* they have to find consensus about the attack time while only communicating via oral messages. However, some of the generals may be traitors, intentionally misinforming other generals about their attack intention. The loyal generals need to have an algorithm in place that ensures that all loyal

generals agree on a common and correct plan of action, while dealing with traitors that do anything they want to hinder the loyal ones from reaching consensus.

Each general observes the enemy and communicates his attack/retreat recommendation to all other generals. Each general therefore has a set of opportunity, provided by all other generals, and bases the decision whether to attack/retreat on a previously defined protocol, like deciding based on the majority. While this works under ideal conditions (no traitors are in place) the algorithm fails as soon as traitors send different pieces of information to different generals: Some loyal generals may receive only "retreat" messages from the traitors, while the others receive "attack", preventing the loyal generals from finding consensus. A loyal general can not rely on the information given by a general directly, since it may be malicious. However, in order to find consensus, all loyal generals must have a set of information that will lead to a globally uniform attack/retreat decision.

The problem of how generals communicate their message can be restricted such that everybody has the same set of information. Since in a computer environment, it can be assumed that every message is delivered correctly, it can be identified who sent it and absence can be detected. For simplicity reasons, whenever a message is absent, a default value, a *retreat*, is used. Split the generals into commanders and lieutenants, while under each commander, there are $n-1$ lieutenants, where n is the number of generals in total. All loyal generals must obey the same order, and a loyal general must send the message in a way that ensures that all loyal generals trust his order. These two conditions are called the "interactive consistency conditions" [44].

Indeed, there is an algorithm that solves the Byzantine Agreements problem with tolerating at most f traitors with $n = 3f + 1$ generals in total [44]. The commander tells its $n-1$ lieutenants about his plan, attack or retreat. For each lieutenant i, let v_i be the value general i received from the commander. Now, lieutenant i sends the value v_i to the $n-2$ other lieutenants. This way, the message propagates redundant to all lieutenants, with each lieutenant not only receiving the commanders message directly but also as forwarded information from any other lieutenant. For each i and each $j! = i$, v_i is the value lieutenant i received from lieutenant j. This message delivery network creates a state of complete information in which every node knows what node sent which message to whom. With the information about what message node i sent to all the other nodes, each node can decide whether node i is potentially malicious, e.g. sent different messages to different users.

With using the majority value for all received v_i, ... $v_{\mathrm{n\text{-}1}}$, it can be guaranteed that each general has sufficient information, and therefore reach consensus among all loyal generals. To this point, it is still unclear how many nodes may be malicious while consensus can still be reached. Clearly, it has to be less than 50%, since otherwise the malicious nodes could reach consensus on their own. Moreover, it can only guarantee that if less than 1/3 of the nodes are malicious. If one third were malicious, and there was a 50/50 disagreement between the honest parties, or if by any reason the honest parties are split and unable to talk to each other, the 1/3 malicious nodes could reach 2/3 of the population votes with one half of the honest nodes, and 2/3 with the other half, destroying consensus. This implies that less than 1/3 of the population may ever be dishonest.

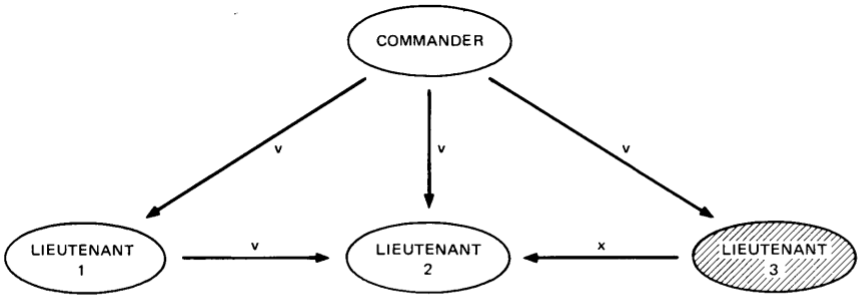

Fig. 3. Algorithm OM(1); Lieutenant 3 a traitor.

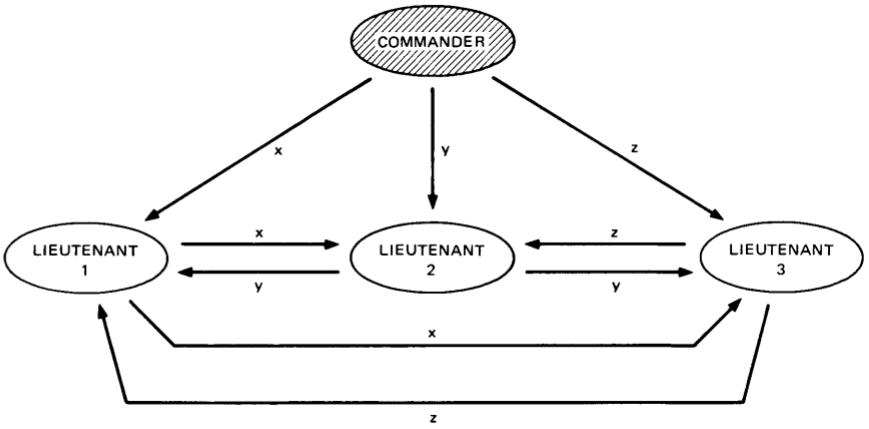

Fig. 4. Algorithm OM(1); the commander a traitor.

Figure 1.1: *Byzantine generals problem with three loyal and one malicious party. If a lieutenant is a traitor (first image), the other lieutenants easily recognize him since he provides values different from the commander (x instead of v). If the commander is a traitor (second image), he is identified since the lieutenants get different messages from the other lieutenants about what the commander said (x, y or z).* [44]

However, [44] also states in the conclusion that the proposed algorithm is expensive in both time and number of messages sent, making it an inappropriate choice for decentralized systems that invoke many nodes that participate in the consensus finding process. Traditional Byzantine agreement is therefore heavily bandwidth- and latency-limited, with having exponential communication complexity with respect to the number of participating nodes [46]. Byzantine agreement further suffers from the problem of Sybil attacks [41], where nodes act as multiple instances and therefore increase their voting weight in the system. Such frauds are typically overcome by having a central list of participating nodes with a central authority that decide whether a new node can register itself for participation in the consensus finding mechanism or not.

1.3.1 Practical Byzantine Fault Tolerance

In the year 1999, a new and more practical algorithm based on Byzantine Agreement was developed: Practical Byzantine Fault Tolerance (pBFT) [49]. The fault-tolerance remains at $m = 3f + 1$, where f are malicious nodes, with guaranteeing liveness and safety in the network. While safety ensures that all operations are either processed or reverted by all nodes together, liveness guarantees that the network won't stop processing new transactions even when the elected leader is Byzantine, e.g. behaving arbitrary or malicious.

The pBFT protocol is used by several modern cryptocurrencies, including NEO, an Asian competitor to Ethereum. While NEO made some modifications to the algorithm, they follow pBFT to a large intend but with using different naming conventions and additional features like delegation. For simplicity reasons, the classical pBFT naming conventions are used.

pBFT distinguishes two types of nodes: primary nodes and backup nodes. A single primary node acts as the leader, while all other backup nodes can switch out the current primary if it seems Byzantine through a view change mechanism. In each view, only one node is the primary, and all nodes seek for consensus for one single view. If consensus is found, the view ends and a new primary is chosen.

pBFT ensures integrity and authenticity of any message sent in the network via applying a cryptographic hash to each message sent, including the message and the sender nodes signature. Furthermore, all new requests are numbered sequentially using a timestamp such that each request can only be executed once and in chronological order.

Every node that participates in the consensus protocol has a record table including the current consensus status. Each view has its own record table, while a view is one process of finding consensus. If nodes reach consensus right away, only one view per set of transactions is needed, but multiple views may be needed if consensus finding turns out to be difficult. Views are labeled sequentially, starting from 0, and bring a form of synchronization into the network. Nodes always search consensus in the same view, and if consensus can not be reached for too long, the view is discarded and a new one is created.

Besides the views, the nodes are numbered as well, starting from 0 to $n - 1$, with n being the total number of nodes that are included in the consensus mechanism. In each view,

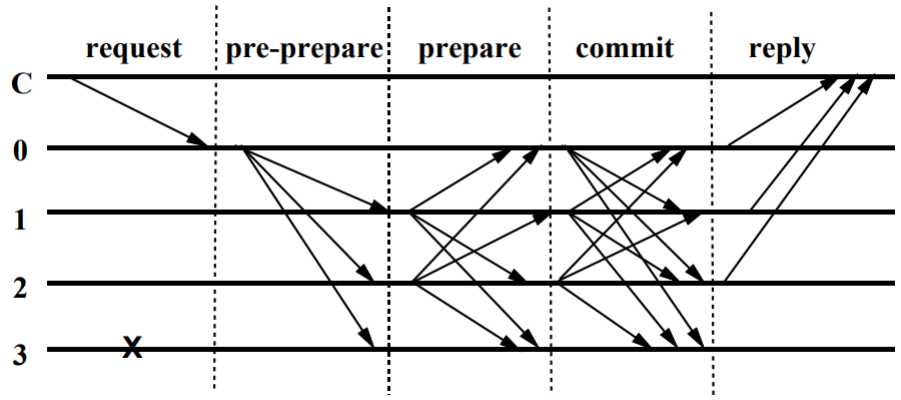

Figure 1.2: *Message broadcasting between Primary (C) and backup (0 - 3) nodes in pBFT, with node 3 loosing connection in the process.* [49]

one node is chosen to be the primary node. At the end of the view, the view number is increased and a new view is generated if no consensus could be found, or a new block is generated if consensus was found, e.g. $n - f$ signatures of backup nodes were reached. If a new block is created, the view number drops to 0 again and a fresh view is started.

When a new transaction is being started, the primary node broadcasts the transaction to the entire network, including the sequence number, the message as well as digest [35], [25]. This phase is called the pre-prepare phase. The backup nodes receive and store these transaction data in their memory if the signature is valid and the message has not yet been seen. Before sending a response, backup nodes validate the transactions and abandon the consensus protocol if invalidity is found, aborting the current view and creating a new one. The view also changes if no consensus was found for a too long period of time. The backup nodes respond to a valid transaction with a broadcasted prepare statement, showing that they will be prepared as soon as they received at least $2f$ other nodes respond to the same transaction with a prepare message as well. When $2f + 1$ non-Byzantine nodes are prepared, the network has a committed state. All committed nodes then broadcast the "commit" message, and as soon as node i has received again $2f + 1$ such commit messages, node i is committed-local. A network that contains committed-local nodes will always become committed as well, ensuring that the transaction will be accepted. The view is finished, consensus is reached.

pBFT is one of the first usable Byzantine Agreement algorithm, ensuring consensus despite Byzantine failure of nodes. It reaches consensus fast and energy efficient, while having decoupled trust from resource ownership. However, all nodes involved must agree on a list of known participants in the network that they include in information distribution. Furthermore, if anybody could join this list, attackers could join multiple times in so-called Sybil attacks, representing many nodes at once and increasing their power inside the network. Furthermore, even though pBFT reduces the bandwidth complexity from exponential to quadratic, a distributed network with a large number of nodes running on pBFT would still quickly run into scalability issues.

Today, blockchains like NEO that are running on practical Byzantine Fault Tolerance

Consensus algorithms are highly centralized, with NEO only having seven consensus nodes in total and five of them being run by the NEO foundation itself [52]. Even though NEO offers plans to include new consensus nodes run by third parties in the future, the limits of pBFT will hinder any blockchain based on this consensus algorithm to reach a true decentralized state.

1.3.2 Federated Byzantine Agreement

The Stellar network uses its own adaptation of Byzantine Agreement called the Federated Byzantine Agreement (FBA). While the consensus mechanism is similar to pBFT, Stellar ads quorum slices to the protocol [59]. A quorum is a set of nodes that is large enough to reach agreement in the network. Quorum slices are a subset of a quorum that can convince a node to agree to a certain opinion. Quorum slices are typically significantly smaller than a quorum. If all nodes belong to the same quorum slice, FBA would be the same as non-federated Byzantine Agreement. While normal Byzantine Agreement requires all nodes to agree on the state of the network where each node must be known and verified, in FBA nodes chose their own quorum slices which are the nodes they trust. Companies, Banks or individuals running nodes on the Stellar network can manually choose, based on personal interest or personal feelings of trust, whose institutions they want to trust and therefore add to their quorum.

In a good network state, quorum slices overlap and form quorum intersections. These overlaps are necessary to ensure that no two disjoint quorums can both reach consensus within each other, but a different consensus throughout the quorums: Quorum A could agree on state X, while Quorum B agrees on state Y. Disjoint quorums can therefore undermine consensus. Nodes must choose what quorum slice they belong to with ensuring to not violate quorum intersection. Nodes must further ensure that slices remain large enough and that the nodes it already contains have something to loose when lying in the consensus finding process.

In FBA, nodes in a quorum slice wait for the vast majority of other nodes in the same slice to agree on a state before considering it settled. Furthermore, these nodes also rely on other nodes which they consider important that may lie in other quorum slices, allowing the network to only reach consensus if the majority of the most trusted nodes agree on the next state. Network-wide quorums arise from individual decisions made by the nodes, while no node has complete knowledge about the whole network, while still reaching consensus network-wide.

Each node has a set of trusted parties. As soon as these parties agree on a new state of the blockchain, the node accepts the state himself as well, reaching consensus quickly in a quorum slice. Each node that depends on other nodes can also be the dependant for other nodes as well. Through this concept, peer pressure arises: A normal node does not fully commit to decision but rather states its opinion. When enough of the other nodes in the same quorum slice formulate a different opinion, the peer pressure forces the node to accept this opinion as well if all quorum slices of the node have an other opinion. An opinion can never be changed on self behalf but only on peer pressure.

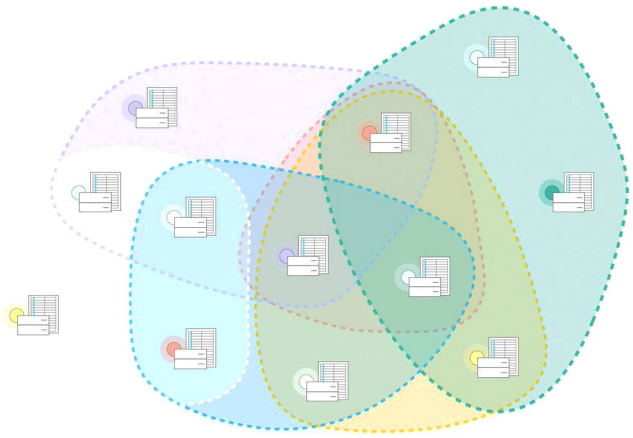

Figure 1.3: *Visualization of Stellar quorum slices among trusted nodes, run by different corporations.* [59]

In federated BFT [22], the voting process for a new decision consists of 4 phases: Initial Voting, acceptance, ratification and confirmation. In the initial voting phase, each node specifies its personal opinion about what is the correct vote. A set opinion can not be changed on self-behalf, but a node can change its opinion if enough of the trusted nodes in its quorum slice have another valid opinion. In the acceptance state, each node make a final decision based on other nodes opinions. A node only accepts a decision if it is not contradicting with its own opinion, or if a vast majority of trusted nodes voted for the other opinion. In the following ratification phase, all nodes from a quorum slice accept one statement. Finally, the confirmation phase is a network-wide agreement on the same decision. It is reached if a sufficient amount of accepting statements is received. The confirmation phase broadcasts the confirmation decision over the network, eventually leading still unsure nodes to a decision.

Consensus within a quorum slice is found via a normal Byzantine Agreement such as pBFT, with using PoW to prevent Sybil attacks and securely splitting the quorum slices into roughly equally powerful portions. A committee combines the quorum slices votes into a new block in the blockchain.

Since quorum slices come to agreement on their own, the protocol has to overcome the risk of liveness: nodes or slices blocking the whole quorum form finding agreement. The protocol neutralizes blocked statements when they risk blocking the consensus finding process using ballots, referendums to the values being voted on. Each node can vote on either committing or aborting any ballot. If a quorum slices is stuck, the nodes can vote on ”commit value X” or ”abort value Y”. Committing a value will automatically result in consensus, while aborting a value leaves the room open for a new ballot but with less opinions.

According to Stellar’s own whitepaper, FBA is the only provably safe consensus algorithm that provides Decentralized Control where everybody can participate in the network without a central authority, low latency that finds consensus within all nodes in at most a few

seconds, flexible trust where nodes can decide on their own which parties they trust, and finally asymptotic security that allows for tweaking the protocol to protect against third parties with large computer power.

The Stellar Consensus Protocol (SCP) addresses the largest downsides of normal pBFT. Since anybody can decide which parties to trust, SCP not just ensures flexible trust but also a much lower latency and asymptotic security since quorum slices are a magnitude smaller than the set of all nodes. Anybody can join the network and choose its own trusted nodes, making SCP a permissionless blockchain.

1.4 Proof of Work

As mentioned in section 1.2.1, PoW is a purely computation-based consensus algorithm. This means that a node has to prove that it "worked". The work done in this case is computational work for which resources have to be invested. This work is called mining in a PoW algorithm. A way to ensure that a node has done work is to pose a problem which is proven to not have any shortcuts to it. This way it is possible to estimate an average amount of tries to solve the computational problem of a given difficulty. With this, one can assume on average how much effort in form of computational power has to be invested so solve the problem. Methods for encryption are perfectly suited to be used for such a problem, since their whole purpose is to encrypt something in a manner which requires an extraordinary amount of computational power to solve while not offering known possibilities to shorten this process. Another key feature of those problems, aside of being hard to solve, is that they have to be easy to validate. As a freshly found solution to the problem is sent to the other nodes, they have to be able to validate the found solution easily. Since the hash inversion is not a decision problem, one can not talk about np-completeness, but the basic principle of "hard to solve, easy to verify" is similar. More specifically the miners need find a solution to a cryptographic hash function of the following form:

$$H(b.n) < d \tag{1.1}$$

where n is the solution nonce which has to be found. Transactions are represented by b which get concatenated to the nonce. H is the hash function of the respective blockchain. In the case of Bitcoin, SHA-256 is used [43]. There is a threshold d which depends on the difficulty of the problem. In a PoW blockchain, all miners try to solve the hash problem given the current latest block, starting with the genesis block. Once a miner found a solution, a block is added to the blockchain and the solution is broadcasted to the network. When the other miners receive a proposed solution which contains more accumulated computational work done that their current blockchain, they don't search for solutions to their problem anymore but try to validate the received solution. If they agree with the solution, they immediately start mining the next block on top of the updated chain.

Forks describe the situation when a blockchain is split into two leading chains. The blockchain now exists as two different version in the network. The first part up to the fork is identical among the network. After the fork, there are two versions with valid

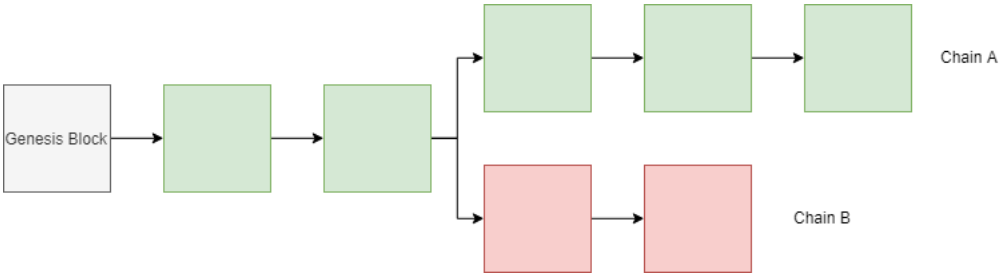

Figure 1.4: *The main chain is represented by the green blocks. The red blocks represent a two block fork which eventually got replaced by chain A.* [64]

solutions to the last common block present in the network. Such a fork can occur if two miners find a valid block at approximately the same time and broadcast it to the network. Because of the delayed information propagation in a blockchain network, some nodes will validate chain A and some nodes will validate chain B. A forked chain is a situation that wants to get resolved back into one single accepted chain. In a PoW algorithm, the longest chain always wins an this can be used to resolve the forks. When a miner starts mining for a new block on top of chain A it puts its computational power behind this chain. Eventually, one of the miners will find a new block and send its version to the network. If this happens to a miner who is appending to chain A, this chain will contain one block more than chain B where no additional block is found yet. When a miner who accepted chain B receives chain A with a block more, it will accept the longer chain as the correct one and will start mining on top of chain A. It is possible among the nodes which accept chain A that a new block is found at the same time as a new block is found among the nodes who accept chain B. This case is called a two block fork. This situation is feasible, but because the possibility to find a new block has only a certain chance, the probability of both chains growing at the same rate will decrease exponentially and one chain will eventually come out on top. In practice, the longest chain is implemented as the chain with the most amount of computational work put in to it. Each node can calculate the computational effort required to build the chain all the way back to the genesis block and compare different chains this way.

Computational power represents a resource which can be used to append a block to the blockchain. Naturally, miners need to be incentivized to spend computational resources. Most commonly the miner can reward himself with an amount of a cryptocurrency defined in the consensus protocol by adding a block to the chain. The amount of resources spent combined with the consensus protocol is the core part to protect against attacks and manipulation attempts. If a miner tries to reward himself with an amount of coins surpassing the amount defined by the shared consensus protocol and then invests all the resources to find a new block, the other nodes will reject the block and all the resources would have been wasted.

Despite the consensus protocols, blockchains can still be vulnerable to different attacks. The double spending problem occurs if a specific coin of a currency is spent multiple times, which can happen if a fork occurs. The currency can be spent for each branch of the fork. To prevent this, usually a certain number of confirmations is awaited to minimize the risk of double spending [8]. To spend a coin a second time on a fork of a PoW blockchain, one

has to possess more computational power than every node who works on the main chain summed together such that a deliberately created fork can be validated. If the attacker does not have 51% of the computational power, the work put in to create the fork will not be rewarded therefore, rendering the attempt useless. Such an attack is called a 51% attack [57].

A 51% attack is a case of a Sybil attack. The goal of a Sybil attack is to create a large enough number of identities to gain a large enough influence to dominate the system [41]. In the case of PoW, the influence is represented by computational power.

By attempting a selfish mining attack on a PoW blockchain, the 51% threshold can be lowered down to 25% [57]. When attempting selfish mining, the attacker withholds found blocks to accumulate their advantage over the rest of the miners. By selectively releasing the found blocks, the rest of the system is unable to catch up and is constantly wasting resources on blocks which in reality are already stale [6].

Compared to other consensus algorithms, PoW protocols usually have a smaller throughput. The throughput of Bitcoin stands at a maximum of 7 transactions per second [29]. As [4] show in their work, a throughput of more than 60 transactions per second can be achieved by adjusting the block size and interval.

1.5 Proof of X

Many other consensus algorithms emerged after the PoW algorithm. All of these "Proof of X" (PoX) consensus algorithms try to eliminate the problems of PoW by introducing a variation to Nakamoto's original protocol. One of the more popular introduced alternatives is the Proof of Stake (PoS) algorithm, which is not entirely problem-free. Similar to PoW, many alternatives to PoS have been published trying to eliminate its problems.

Thus, it's only natural to compare such PoX algorithms to either PoW or PoS and to examine how they have improved over them, which vulnerabilities and problems could not be eliminated and which new vulnerabilities have arisen with the new proposed consensus algorithms. This will be done in this section, but to get a better understanding of these consensus algorithms, PoS needs to be introduced and compared to PoW in a first step.

1.5.1 Proof of Stake

The goal of PoS is the same as of PoW: To reach a consensus over who is allowed to add the next block to the chain so that the state of the distributed ledger is identical across all nodes [63]. However, as seen in the previous section, the PoW algorithm has many underlying issues. The main issue that the PoS Algorithm is striving to solve is the excessive usage of resources due to the computation-bound principle of the PoW algorithm - that is to say, due to the energy consumption through mining hardware [24][64].

A user in a PoW-based blockchain is *mining* the next block by allocating processing power. Similarly, a user in a PoS-based blockchain is *minting* the next block by using, depending

on the implementation, some or all of his own cryptocurrency [8][64]. The amount of cryptocurrency the user is willing to use for that purpose is called *stake* [64].

In PoW, the first user who solves the given inequality is the one who appends the next block to the blockchain. The higher the processing power or the more processing hardware a user possesses, the higher the chances of him mining the next block [64]. In contrast, in PoS the user who appends the next block is randomly selected. The chances of being selected is proportional to the amount of stake (in relation to the total amount of all cryptocurrency in the chain) the user possesses. If a user possesses 1% of all the circulating cryptocurrency in the blockchain, then his chances of minting the next block are 1% [8][63].

The PoS inequality that needs to be solved by minting can usually be generalized as the following formula:

$$h(a_1, ..., a_n) < s(b_1, ..., b_n) * d \tag{1.2}$$

where

- $h(\cdot)$: Hashing function,
- $a_1, ..., a_n$: Input arguments for the hashing function,
- $s(\cdot)$: Some sort of a function that involves the stake,
- $b_1, ..., b_n$: Input arguments for the stake function,
- d: Difficulty constant.

For the simplest case, assume the function $s(\cdot)$ would return the stake only. Since d is a constant, the difficulty of solving the inequality would depend on the amount of the stake.

PPCoin, one of the first or even the first blockchain to introduce the PoS consensus algorithm [19][64], uses a concrete example of (1.2) [8][34][64]:

$$h(B_{previous}, A, time) < s(bal(A), age(A)) * d \tag{1.3}$$

where

- B_{Previous}: Some data of the previous block,
- A: Address of the stakeholder,
- $time$: Current time in seconds; usually in UTC,
- $bal(A)$: Balance in the stakeholder address,
- $age(A)$: Coin age.

Firstly, it is apparent that *time* is an additional factor that influences this concrete inequality. Secondly, it is important to understand that the time unit being seconds restricts the hashing attempt of a minter to 1 attempt per second. A certain deviation interval is set for UTC. For example, *time* can deviate ± 1 hour from the current UTC time. Since all the other parameters of $h(\cdot)$ are not changed as often, this would limit the amount of possible arguments of the hash function to 7200 [8], making the PoS consensus algorithm effectively more power efficient than the PoW algorithm due to the bound domain space [62].

It is noteworthy that the *time* parameter could lead to a "Timedrift Exploit", where an attacker could calculate the hashes into the future to predict the probability of a certain user to mint the next block, effectively making it a long-range attack (See Section 1.5.1.2). However, this issue has been already solved. For more details please refer to [34]. Another detail of PPCoin is the initial use of PoW algorithm to distribute the first coins. Using PoS from start on without an Initial Public Offering (IPO) would not only increase the chances of a Sybil attack but it would also have economical impacts, like empowering early individual participants with a first-mover advantage [62].

1.5.1.1 Comparison to PoW

Both algorithms provide an incentive in form of cryptocurrency to motivate the users to participate in the consensus process [8][26]. However, since the domain space of the hashing function in PoW is infinite, the mining process becomes an "arms race": Each individual is competing against each other by trying to find ways to solve the inequality in a faster way [8]. While the choice of a memory-hard hash function can lower the cost-advantage of ASIC's [64], the computation-bound design principle of the PoW algorithm cannot be fully eliminated. On the other hand, that computation-bound design is responsible for the strong security and resistance against eventual malicious attacks. Therefore, the trade-off between the two algorithms occur in terms of power-efficiency vs security.

More particularly, the security of PoS algorithms is by default so poor [47] (See Section 1.5.1.2) that it is necessary to come up with multiple ways of securing the algorithm against malicious attacks. Since almost no computational power is needed, the chances of having centralized resources diminishes in PoS. The decentralization (as defined in [69]) of PoS is therefore improved over PoW. At the same time, the disengagement from computational power allows for a faster throughput. In terms of performance and scalability, PoS implementations can vary widely. While the hybrid PoW/PoS algorithm of PPCoin has an estimated throughput of 8 transactions per second [54] in comparison to Bitcoins 7 transactions per second, Ouroboros can achieve a throughput of 256.7 transactions per second (See Section 1.6). While PoS shows some improvements over PoW in regards to scalability, better scalability and security at cost of centralization can be achieved by a Delegated Proof of Stake (DPoS) algorithm (See Section 1.5.1.3).

1.5.1.2 Problem and Vulnerability comparisons between PoW and PoS

Both algorithms are potentially vulnerable to Sybil and 51% attacks. While the resistance to Sybil attacks in PoW comes from the needed computation power, the resistance against such an attack for a PoS system comes from an economical perspective: stakeholders with a high amount of stake are interested in keeping the network secure, since they don't want to sabotage themselves by attacking the system and risking lowering the value of their stake [34].

There are multiple ways to perform a double spending attack. The first method is with a long-range attack. If a PoS System is new, not yet established and the genesis block has just been created, a long-range attack can occur even if a stakeholder owns 1% of the total available coins [66]. At later stages, a higher stake amount might be needed [34]. In a first step, it is possible for an attacker to start minting on a forked new chain that is not the main chain. He can then spend some of his coins in the main chain and start minting on top of the forked chain in which he didn't spend the coins. In a next step, he can start minting into the future by executing a lot of calculations on the forked chain. Since he still owns 1%, the attacker will be faster in generating new blocks than the other minters in the main chain. When his chain gets validated, he will be able to spend his coins a second time [2][66][34]. There are multiple solutions to this problem, PPCoin implements two solutions to solve this problem. First, the input argument $B_{Previous}$ for the hashing function changes every 6 hours to make it impossible to foresee the future [34]. This solves the problem for both phases, when the chain is not yet established and when the chain is already established. And second (an indirect solution), PPCoin started using PoW and then slowly switched to PoS [62]. This especially solves the 1% attack. The second solution also solves the initial distribution problem of PoS, where some stakeholders would have too much influence over the system due to the first-mover advantage. Starting with a PoW algorithm is more competitive and certain miners do not possess more influence than the others [8][62].

The second way of performing a double spending attack is tied to the "nothing-at-stake" problem. Since there is only a limited amount of possible inputs for the hashing function of the PoS algorithm, forks are more likely to happen [34]. In case of a fork, a rational individual in PoW would continue to mine on only one forked chain without splitting its computation power, since that decision leads to the highest Expected Value (EV), as seen in figure 1.5 [68]. In case of a fork in a PoS system however, a minter can decide to participate on any arbitrary number of forks without any penalty, since PoS is not computationally demanding and since there are no other (monetary) punishments. As seen in figure 1.6 [68], the minter increases his chances of being on the chain that will be chosen as the main chain in the future. The stakeholders EV increases accordingly. In this context, if a stakeholder decides to participate in a fork, this participation can be interpreted as a "vote" [26]. If enough participants act rationally by trying to increase their EV, they will vote for multiple chains and since they have voted for multiple chains, no consensus can be achieved [26][34][68]. Not only does it disrupt reaching the consensus, but this problem also facilitates a double spending attack. Similarly to the first method, a stakeholder with enough influence can effectively start minting at two forked chains, spend his coins at one of the chains and then leave that chain to mint on the other forked

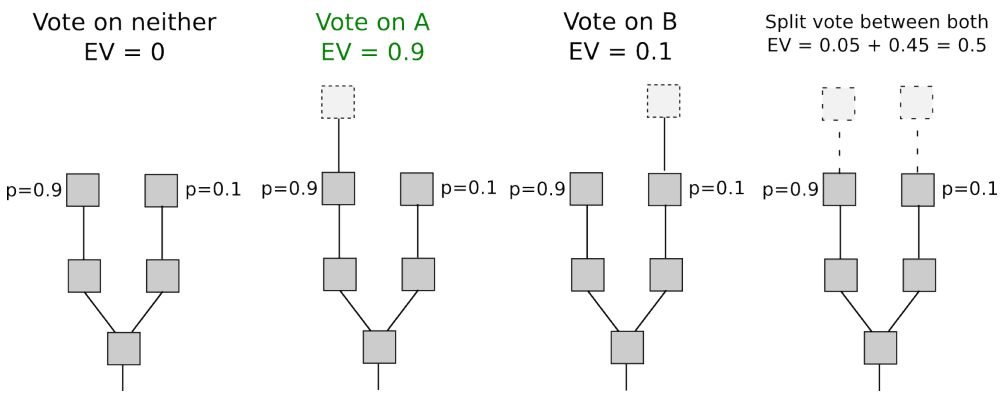

Figure 1.5: *Shown are the 4 possible scenarios how a miner could act in case of a fork in a PoW system. The Expected Value "EV" changes depending on the miners choice. The green marked one is the rational choice.* [68]

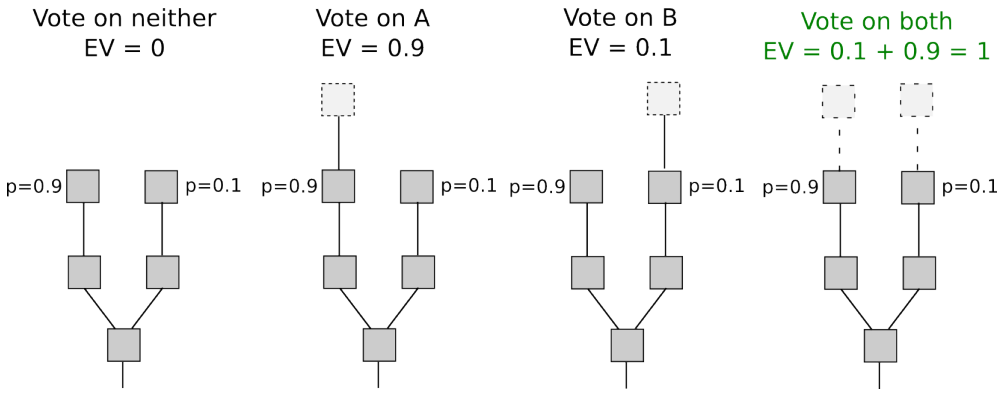

Figure 1.6: *The same scenarios as in figure 1.5, but this time in case of a fork in a PoS System and with the highest Expected Value on the* 4^{th} *scenario.* [68]

chain where the coins are unspent. Since that increases the probability of generating a longer chain on the second fork, the attack might succeed and the attacker could spend his coins a second time [39]. The nothing-at-stake problem can be easily solved by a DPoS algorithm [34][64]. See Section 1.5.1.3 for further details.

The third method to double spend is enabled through a bribe attack. For more details refer to [34]. A selfish mining attack seems to be possible in both [30][38]. In PoW, the selfish mining can waste the resources of other miners. Since there is no heavy computation involved in PoS, this is a non-issue. Selfish mining is used to increase the attackers own reward relative to the other participants rewards. This also applies to PoS and the result can even be worse in PoS systems [30]. Depending on the implementation, a grinding attack can also occur in a PoS System. In case of PPCoin, a minter might try to use different input arguments in such a way that the likelihood of him being selected increases [68]. This list of vulnerabilities is not complete. The amount of security issues with a standard PoS algorithm is large. For more types of attacks, refer to [2], [8] and [34].

1.5.1.3 Delegated Proof of Stake

Many of the sources used in this section are not official or not formal, as there don't exist a lot of them. Some of the issues of PoS can be overcome by a DPoS consensus algorithm while keeping the power efficiency achieved by PoS [11]. DPoS is based on a delegates system. Instead of minting by themselves, the stakeholders of a DPoS System can vote to select delegates who then will mint the new block, effectively centralizing the block generation. A stakeholders voting power is proportional to their amount of stake. Since the delegated users have (temporarily) all the control over the chain, there would be nothing that would hinder them from misusing the nothing-at-stake problem from the PoS algorithm to double spend [8]. That's why it's especially necessary in DPoS to implement a method to punish users that are misbehaving. Casper for example, the soon to be implemented PoS protocol of the Ethereum blockchain, punishes users who are trying to mint on multiple blocks by removing the malicious users complete stake [64]. In BitShares, a DPoS based system, the misbehaving block-producers can simply be "voted out" of the system [11].

The DPoS-based blockchain BitShares will be used as a concrete example. Based on self-governing democratic principles, it uses a *witnesses* and *delegates* system [11]. Both, the witnesses and delegates (also called committee) need to be selected by the stakeholders (the *workers* need to be selected too, but are of no relevance for this comparison) [11]. The amount of the witnesses and delegates is also determined by the stakeholders, depending on the wished decentralization level [10]. The more stake a stakeholder possesses, the more weigh is assigned to his votes and he can split his voting power amongst the witnesses and delegates. It's possible for a stakeholder to proxy there votes instead of directly choosing the witnesses/delegates, meaning they can lend their voting-power to another stakeholder that will then perform the voting for them. Witnesses are the block-producers of the blockchain and they get rewarded for their work [11]. They are socially pressured to distribute as much of their rewards as possible to the voters, otherwise they would be replaced by other witnesses [65]. The elected witnesses are sorted in an ordered list. A time slot gets assigned to each witness during which they have to produce the next block.

After the end of the list is reached, the witnesses get shuffled into a different order and the block-producing is repeated again and again. The committee members on the other hand are responsible for signing the blocks produced by the witnesses and changing the parameters of the blockchain, like block size and witness fee [10][11]. They do not get rewarded for their work [10].

The scalability and security are improved by increasing the risk of centralization [11][53]. In theory, according to the inventor of the DPoS algorithm and inventor of the BitShares and EOS protocol (also DPoS based), Daniel Larimer, it should be possible to reach a throughput of 10'000 transactions per second without optimization, 100'000 transactions per second with optimization [11] in BitShares and even 1'000'000 transactions per second in EOS [31]. In practice as of now, the all time high of BitShares in a test-environment was 3328.33 transactions per second [5] (claim could not be verified as source is not public) and 3996 transactions per second of EOS on a live environment. EOS is currently moving between 50 - 90 transactions per second [20] while BitShares is moving between 7 - 10 transactions per second [9]. The throughput can potentially reach higher numbers compared to many PoW and PoS implementations. If such high amounts are sustainable over a longer period of time in a real environment or if a higher throughputs can be reached in practice has not yet been proven.

Further on security, the nothing-at-stake problem is eliminated by design. Additionally, as a security measure in BitShares, the chain with the highest witness participation rate will be chosen as the main chain. This does not only improve the network stability in cases of latency and "communication breakdown" but also hinders a malicious witness from trying to build a fork, as he cannot increase the participation rate in his chain [10]. Since no alternative chain can be build successfully, a selfish mining attack is not possible. Compared to many PoS implementations, a long-range attack in BitShares can be performed if more than 67% (a BFT problem) of the witnesses with 51% of the total voting power cooperate to perform it [8][21]. This attack is more probable if, as in a real world representative democracy, less voters participate in the election system and if they vote by proxy. The witnesses could also corrupt the proxies to get more votes, performing a bribe attack [65]. In contrast to PoS implementations, the attack cannot be performed if a single voter owns 51% of the total coins due to the 2 separate delegate-type system [21]. However, a stakeholder can influence the system by owning more currency than the other voters [65]. For a grinding attack to happen, a witness must be able to influence his chances to be selected more often than the other witnesses per period. This is not possible by design in BitShares. Also if a witness would try to cheat the system through fake network breakdowns and by increasing the latency, the algorithm would simply skip him and the stakeholders would vote them out for not behaving honest [10].

1.5.2 Proof of Elapsed Time

Another PoX algorithm that has been developed by Intel is called Proof of Elapsed Time (PoET) [45]. Generally, the way a PoET consensus algorithm works is simple: a user gets assigned a certain wait time. After the wait time, the user is allowed to produce the next block. For the user to be able to prove that he has waited the assigned time and that

he does not cheat, a *trusted hardware* is needed [64]. Trusted hardware provide a safe environment in which certain instructions can be executed without getting disrupted by the user or by a malicious attacker [45]. Some Intel CPUs are in possession of the "Intel Software Guard Extensions" (Intel SGX), a set of instructions that allow applications to run in a trusted environment [70]. This is done by loading the data into enclaves in the memory [37]. The PoET algorithm can be run in such an enclave, thus preventing cheating and allowing the users to prove their wait time [57]. To further increase the security, there needs to be a central authority like Intel that maintains a list of trusted certificates that can check if the CPU is trustable [64]. Sawtooth Lake is a concrete blockchain-technology, also developed by Intel, that utilizes Intel SGX [36] and that can be used in a permissionless or permissioned network [70][32]. Concretely, Sawtooth Lake works by assigning a different random wait timer to each user. The wait timer starts counting down and the user with the smallest wait time left is chosen as the next block-producer [64].

Since almost no computation needs to be done, it's as demanding as power efficient as for example PoS or DPoS [70]. On the downside, a user can still increase his chances by buying multiple CPUs, making it an arms race like in PoW again [57]. This is the so called "stale chip problem" [57]. The throughput can go up to 1000 transactions per second [17] (claim could not be verified) and further scaling is possible. This kind of scalability is achieved by giving up decentralization and some security. Decentralization is essentially given up almost fully, since there needs to be a trusted authority like Intel [32]. Thus, it might be reasonable to use PoET in a private setting, where trusting a certain authority and decentralization are seen as less risky [32]. The centralization is a double-edged sword in regards to security: Since the application can run in trusted hardware, certain rules can be enforced to the user, like always running the newest version of the protocol [64]. This adds a lot of security to the blockchain, essentially making it resistant too many types of attacks. However, the trusted hardware is also its weakest point. If the trusted hardware has a security risk and gets exploited successfully, the malicious user could cheat and that would enable all sorts of attacks like double spending, selfish mining and grinding attacks [45]. According to Intel, this is a non-issue since it can be easily detected with a statistical analysis [57]. Other than that, the security risks of a PoET algorithm are largely unknown [45]. A successful attack can occur, if $\Theta(\frac{log\, log\, n}{log\, n})$ nodes, where n is the total number of nodes in the system, are compromised. This is different compared to the concept of a 51% attack, since it depends on the size of the system. The system is less vulnerable the bigger n is [45].

1.6 Ouroboros

The following section adds a novel and interesting approach to the consensus algorithms already mentioned. Ouroboros is a provable secure algorithm that uses PoS in specific timeframe and timeslot architectures with a protocol that provides an unbiased randomness to elect block generators, named slot leaders.

Ouroboros is a PoS algorithm used by Cardano which is a public decentralized open source blockchain and cryptocurrency project [13]. The development team of Cardano consists of

a large collective of researchers and experts. Unlike many other open source projects Cardano set the focus on mathematical provable correctness and security through reviewed research of academics and developers. The modular structure is open for functionalities like delegation, sidechains, subscriptable checkpoints, efficiency for light clients, different forms of random number generation and even different synchronization assumptions [14][15][16]. With this flexibility and a research driven approach, Cardano aims to stay flexible, future-proof and with a test setup with 40 nodes and a slot frame of 5 seconds a median of 257.6 transactions per second could be achieved [2].

1.6.0.1 Basic Model

As in PoS, nodes with a positive stake may participate in running the protocol. An elected slot leader can generate new blocks through listening to transactions, generating the block, signing the block with the secret key and finally publishing it to the network. In Ouroboros, time is divided into epochs which are again split up into slots. A slot leader is similar to a miner in Bitcoin, the election process however is different. Each slot has only one leader which has a right to generate exactly one block within that time slot. It follows that because in each slot only at most one block can be generated, in each epoch at most n blocks can be generated, where n defines the maximum amount of slots. If the elected leader misses their slot, the right to generate a block is taken away until the same leader is elected again to generate one block [2].

The two principles, persistence and liveness as described below, as well as a robust transaction ledger, given that the maintaining ledger is divided into time slots which determine the order of transactions, are embedded in the ledger. These factors together provide a robust transaction ledger where honest transactions are adopted and become immutable as soon as the depth of the block is more than k blocks.

Ouroboros reaches persistence and liveness through stability. A transaction is seen as stable when the depth of the block is bigger than the security parameter of k blocks. So when it holds that one node agrees on a transaction to be stable, all other honest nodes agree on that transaction as well, persistence is achieved. If any honest node however would not agree upon a transaction, persistence would not be given. Bitcoin's measurement is comparable to the idea of persistence described where the longest chain of blocks is considered as the correct chain [56].

In Ouroboros, a transaction ledger reaches liveness when a transaction is accepted and added to all nodes after a certain amount of u timesteps. If the response is honest and confirmed by all nodes then the transaction will be reported as stable and the valid transaction will be added to the blockchain. In comparison to Bitcoin where an honest majority assumption and the common prefix property of the backbone protocol of Bitcoin lead to persistence (Lemma 24, Definition 4 and Theorem 16) and liveness (Lemma 25) [42], with a PoS algorithm this is not achievable in the same manner. In PoW, the competition of mining a block lies outside of the blockchain whereas in PoS the leader generating a block is located in the blockchain. This however leads to the need of countermeasures against grinding and nothing-at-stake attacks which could lead to loss of liveness or persistence.

In PoW, a nothing-at-stake attack would cause to split the resources between every fork which would lead to also split the monetary value earned. So in PoW a natural incentive is given to only chose the longest fork instead of multiple forks. However, voting for multiple forks in PoS does not imply more costs nor a splitting of the reward. Under a rational assumption of an attacker a nothing-at-stake attack in PoS is executed only when there are no penalties in the case when adversaries generate blocks on more than one fork. The consequence is that rational nodes would stake on the original chain as well as on the attacker's chain and if other nodes follow the attacker's chain, the persistence would be violated since the history could be rewritten. Ouroboros proves in their scientific paper that a nothing-at-stake attack would be infeasible [2] since each slot is assigned to a identified slot leader uniquely as well as randomly, and to be elected as a slot leader one must have at least 1% at stake [58]. But even if an adversary would try a nothing-at-stake attack, the chain selection rule used in Ouroboros suggests to ignore deep forks that differ from the last received block.

Grinding attacks or "Stake grinding" are a collection of attacks in which the validator has the intention of being elected to mint a next block more often than the random selection would allow [2][67]. In fact, such a grinding attack can also be used in order to operate a double spending attack. In case a grinding attack succeeds, and the electors could change the random election in their favor, liveness would be violated. In Ouroboros, a multiparty coin-flipping protocol is used to obtain the needed (dynamic) randomness for the leader election process which prevents grinding and double spending attacks [2] in each epoch, whereas in conventional PoS systems the random election is based on raw data. The blockchain itself is used as a broadcast channel where in each epoch a coin flipping protocol is executed. At the moment Ouroboros' slot leaders are declared publicly. There are different variations of Ouroboros as for example Praos which will not be discussed further in which only upon publishing a block, stakeholders are able to detect slot leaders. The committee has the power to determine stakeholders that have the permission to find the next group of stakeholders for the following protocol execution as well as the results of the leader election processes for the epoch and thus allowing that slot leader to add a block to the blockchain.

The multiparty computation (MPC) used in Ouroboros delivers the randomness which is needed for the leader election in every epoch. A MPC is required in a trustless environment where nodes do not need to trust any other node and thus can work on private inputs and messages sent from nodes to nodes are handled as black boxes. In the first step of a MPC a commitment is formed. During this formation the defined slot leaders do the coin flipping and generate the random numbers. Under the two assumptions that the outcome of the coin flipping is guaranteed (guaranteed output delivery) only if honest majority is given and R denotes the time of an epoch which contains $10k$ slots in order to simply relate to, in each time slot the following protocol is executed: In the commit phase which lasts $4k$ slots (Figure 13, Protocol DLS [2]) Stakeholder A generates the randomness and encrypts with the private key each generated share of the secret under the respective public key and posts it to the blockchain. The share cannot be opened at this point and remains a secret value to the other Stakeholders B. If however there is a Stakeholder C which controls more than half of the shares, that secret can be opened through reconstruction. The next phase is called Reveal Phase which also lasts 4k slots. Upon receiving a random secret from Stakeholder B with the same size as the secret from A, stakeholder A again

sends a black box with a key to open the black boxes to Stakeholder B, now containing the key to open the first message that was sent in the Commit Phase as well as the secret. In case Stakeholder B compares the secret received in the Reveal Phase with the opened secret by the received key and equal values are found, the process proceeds. If the opened secret does not match each stakeholder terminates. Finally, in the Recovery Phase which starts at the $8k$th slot and lasts $2k$ slots, the committee checks for stakeholders that have not revealed their secret yet. These stakeholders now submit their shares and as soon as at least half the shares are posted, all secrets can be reconstructed for each stakeholder. Obviously when less than 50% of the participants broadcast in the reveal phase due to either malicious behavior or simply not being online, the protocol can not reconstruct the secrets from the stakeholder and this process can not continue.

With all the secrets that are publicly known, an XOR computation results into a probability that is used as a seed for finding a satoshi, and therefore the owner for each slot in the next epoch can be elected. As soon as the slot leader is found and the next committee is formed, the next epoch starts.

Following the rules of the protocol as a participant which is proven to be an approximate Nash equilibrium is incentivized. Payoffs are offered for protocol action like being a committee member, endorsing a set of inputs or sending messages for the MPC protocol and thus it applies that for those stakeholders that act faithfully, the equilibrium holds when all the stakeholders are rational. This design choice eliminates the need for strong countermeasures against selfish mining and block withholding. However, for not rational attacks and stakeholders with strictly more than 50% of the stakes, this does not hold for Ouroboros since the requirements liveness and persistence are not given anymore. In that case it would be possible that the honest users will be skipped or left out [2].

It is notable that direct comparisons in throughput and performance as well as applicability between Ouroboros and any other already established, under real world conditions tested consensus algorithm can not be done as of yet. Although the research background is rigorously encouraged in Cardano, Ouroboros has not yet been tested in a real world scenario with hundreds or thousands of nodes and thus the scalability under those conditions can not be assessed yet. Nevertheless, Ouroboros is a good example of creating a consensus algorithm with focus on scientifically proving correctness and security. Therefore, Ouroboros is classified between being decentralized and secured, similar to Bitcoin. Due to the experimental testing of the scalability of Ouroboros, further evaluations need to be considered in order to be able to reclassify Ouroboros for a potential solution to the trilemma.

1.7 Open Representative Voting

Another exotic consensus algorithm is the open representative voting (ORV) combined with PoW, DPoS and an asynchronous block lattice architecture where each account has it's own account-chain in order to send and receive transactions as well as calculate it's balance.

Raidblocks launched in 2015 and rebranded in 2018 as Nano is a cryptocurrency which uses an innovative Direct Acyclic Graph (DAG) as an underlying structure and ORV to find consensus. Instead of one blockchain which has to be agreed by the global network like in Bitcoin, with block lattice only the sender and receiver have come to consensus in order to complete a transaction meaning that users are assigned to their own account-chains. Consensus is achieved by a balance-weighted vote of representatives on transactions that are conflicting, comparable to DPoS. PoW is additionally used to counter spamming attacks on the senders side. Considering the PoW sequence this leads to a maximum of 5 transactions per second that can be sent from the same account and about 10'000 transactions per second which mark the theoretical maximum bound of the current algorithm [19] using up-to-date hardware. Also mentionable is that the PoW for the transaction is pre-cached which means that the next transaction will be instantly. Nano's PoS protocol with ORV enables more usability for users as there are no fees involved in processing transactions. As of writing this paper, in comparison to a Bitcoin transaction that needs 400KWh [24] to transact one BTC, one Nano transaction needs less than 0.001KWh. With these features, Nano seeks to revolutionize the cryptocurrency field with fee-less real-time transactions [19].

1.7.0.1 Basic Model

Each account has it's blockchain which represents the accounts transaction history. By only allowing the owner of the account to update it's so called account-chain, the update is immediate and does not depend on other transactions on the block lattice. And since nodes record and rebroadcast, a wide spectrum of devices and hardware can be used allowing potential integration into Internet of Things. Several rather small projects, as for example an implementation of payment to a charging device for smartphones, have been realized[12].

Each transfer requires two transactions where the *send* defines the amount from the sender and the *receive* simply adds the amount to the receivers account. Handling the transfer of funds in this way brings several advantages. The first advantage and also a key design component is that incoming transfers of funds are asynchronous which allows the receiving account to decide in which order the incoming blocks are signed. Keeping the information slim to fit into UDP packets forms the second advantage, although in the future Nano considers to switch to TCP [3]. Because the running total balance is up to date at any time, the associative addition of further transaction amounts is not dependent on the order of the respective transactions. PoW as it can be seen in [7] where a C++ implementation of the PoW algorithm is shown, is only used as a countermeasure against spamming attacks. On high end consumer computers as of 2019, this PoW algorithm will exceed the threshold variable in the while statement in a fraction of a second [19], similar to Hashcash [1].

Incoming transactions can either be settled or unsettled. In settled transactions, the account has already generated receiving blocks, whereas in unsettled transactions, the cumulative balance is not added yet. So this implies that the nodes or their representatives must be online in order to receive transactions [27]. Moreover, during the sending process,

the sender must have a balance and therefore an account with the address must be registered. The *send* message contains the fields previous, balance, destination, work, type and signature. In the field previous a hash of the previous block in that account-chain can be found and in the destination field, the destination of where the funds are being sent to is defined. After a block is being confirmed, the broadcast to the network starts and the funds are pending, assuming that the deduction is instant. Pending funds cannot be revoked by the sender. As soon as the receiver signs a *receive* block the sending process is completed and the receiving process starts. The receiver then creates the *receive* block and adds the hash of the *send* transaction to the source field. The creation of the *receive* block leads to the broadcast and after that, the balance is updated and the transaction is completed [19].

As in DPoS, representatives can be elected in block lattice in order to resolve conflicts by voting. Not every user might want to run a node and giving representatives the voting power might be favorable for the user for usability reasons as mentioned before, where a node must be online in order to receive transactions. At the time of creation of the account a representative must be chosen. The representative can easily be redefined through specifying this in the most common wallets and changed through a *change* request, similar to one of the transaction messages shown before. A *change* transaction consists of the fields previous, representative, work, type and signature. The transaction subtracts the voting weight from the previous representative and adds it to the newly defined representative. Weight of a node is the sum of all account balances that this node is the representative of [19]. In Nano, every node that has a stake of at least 0.1% can become a representative and the incentive to do so lies in not having to rely on a third party [50].

Forks are seen as either bad programming or an adversary double spending of an owner. In an adversary attack scenario double spending would inevitably lead to a fork in which two blocks refer to the same previous block. As soon as such malicious actions get detected, a voting is broadcasted. The representatives weighted nodes will vote for a short period of time in order to find the winning block. Voting weights of the representatives are the sum of all accounts that have delegated their coins to them. After using the DPoS algorithm, the winning block is found as the most voted block and is then kept in the node's ledger where as loosing blocks are removed. Representatives only vote when such double spending attacks or forking occurs. Then if the following conditions are given a transaction is seen as valid: First, there should be no duplicate transactions which means that there should not exist forks like mentioned. Second, each transaction is signed by the owner of that account in the creation of the *send*, *receive* or *change* transactions. And third, the accounts and especially accounts with a sending transaction must have a balance and on the receivers side the PoW threshold must be met.

Other attacks, as for example Sybil attacks where an adversary would create or maliciously conquer a vast amount of Nano nodes, would not lead to more delegational power nor incentive since the DPoS system relies on the delegated sum of stakes and not on the number of nodes. One other possible attack is flooding the network with either unnecessary but valid transactions or transactions where minimal amounts of currency are sent. Both of these types are to a degree countered by the PoW algorithm which limits the sending from an account to the power of the given hardware (Table 1 [19]). A third type of attack is precomputing blocks on the accounts chain with the malicious purpose of

broadcasting and creating a Denial of Service (DoS) attack. Combining this with a 51% attack on the weighted system it would cause the system to break. At the moment, Nano is investigating ways to mitigate such a combination. The design of the consensus system itself would require the adversary to damage their investment since the ledger would be of no value anymore. Also considering that depending of the size of the network at the time of the attack, that investment of the node could not be recovered. Important to note and consider is that a 51% attack can be lowered to for example 33% in the case a DDoS is able to put another 33% of the representatives offline in case of a combination of attacks. The last attack regarding Nano is bootstrap poisoning which might remind of selfish mining. In fact, in bootstrap poisoning, the attacker tries to hold some old accounts private-key for a long time in order to waste time for the honest nodes in the network. By keeping that old representation of the network that adversary representative would try to get votes of nodes by presenting wrong information. Nano solves this problem by letting nodes pair themselves with a database of accounts with legitimate block heads [19]. Since this solution would involve using a database representation of information which should be stored on the blockchain, true decentralization would be questionable in the case of Nano.

Block lattice as the architecture behind consensus algorithms using PoW for spamming, DPoS for conflicts, ORV and DAG consisting of nodes of account-chains offers potential for very good performance. Nano picks each algorithm carefully in order to solve the problem which the algorithm was designed for. At the moment, the representative with the highest weight controls 26.43 % of the stake. In comparison with the leading representative, the second and third most voted representatives control 19.62 % of the votes combined. And although this distribution does not seem quite well, a voting in the form as described has not been needed yet [51]. Therefore, if the DPoS used to find consensus in Nano would be used, decentralization would be questionable as 8 out of 16 of the top representatives are official representatives owned by Nano. On the other hand, scalability in the block lattice architecture is seemingly given. Considering rational users and representatives, Nano with the ORV consensus algorithm can be seen as secure and scalable and thus does not provide a complete coverage for all of the three parameters in the trilemma.

1.8 Conclusion

The following tables serve as an overview of the different features analyzed for each algorithm. Table 1.1 gives an overview of the general features while table 1.2 focuses on the trilemma and the security aspects.

A wide variety of consensus algorithms and organizations that are using these different approaches have been analyzed. As seen in the Byzantine generals problem, not every node can be trusted in a public environment. Using a protocol which is well suited for public environments (*e.g.*, PoW) does not necessarily fit to a permissioned blockchain. Depending on the use case, some algorithms are better suited for permissioned networks, as for example PoET. The public permissionless Bitcoin blockchain, being a permissionless network, solved the Byzantine problem by sacrificing scalability while trying to achieve a high decentralizion of nodes. PoW solves the Byzantine problem by reaching a consensus

Table 1.1: Comparison of consensus algorithms 1

Consensus Protocol	**Working public implementation**	**Incentive**	**public/private blockchain**	**Transactions per Second**
fpBFT	Stellar Consensus Protocol	Securing network	public	1000
pBFT	Neo	Monetary compensatoin	permissioned	1000
PoW	Bitcoin	Monetary compensatoin	public	7 [1]
PoS, PoW	PPCoin	Monetary compensatoin	public [2]	8 [3]
DPoS	BitShares	Monetary compensatoin	public	3328.33 [4]
PoET	Sawtooth Lake	depends on use case; monetary compensation possible	public permissioned private	1000
PoS Ouroboros	Ouroboros	Monetary compensatoin	public	256.7
DAG Block Lattice DPoS, PoW	Nano	not realiant on third party	public	10000

[1] Up to 60 in theory.
[2] In general, an implementation of a PoS algorithm can be private or public.
[3] These values depend strongly on the implementation and not only the consensus algorithm.
[4] On a stresstest only, current transactions per second is around 7 - 10.

Table 1.2: Comparison of consensus algorithms 2

	Trilemma			Resistance to Attack				
Consensus Protocol	**Scalability**	**Security**	**Decentralisation**	**Double Spending Attack**	**Grinding Attack**	**Nothing-at-Stake**	**Selfish Mining**	**Sybil Attack**
fpBFT	✓	✓	X	S	I	I	I	33%
pBFT	✓	✓	X	S	I	I	I	33%
Pow	X	✓	✓	S [1]	I	P	S	51% [2]
PoS, PoW	✓	X	✓	S	P	P	P	1% to 51%
DPoS	?	✓	X	S	S	S	S	67% of witnesses 51% of stakeholders
PoET	?	✓	X	S	S	S	S	<50% [3]
PoS Ouroboros	?	✓	✓	S	S	S	S	51%
DAG Block Lattice DPoS, PoW	✓	✓	X	S	I	I	I	51%

?: Questionable due to quantity of evaluations

S: Secured

I: Impossible by design

P: Prone

[1] 51% is required and its very expensive.

[2] Selfish mining by colluding nodes can lower the security threshold down to 25%.

[3] The percentage is variable, since it depends on n, the amount of nodes in the system. The formula to calculate the percentage is: $\Theta(\frac{log\,log\,n}{log\,n})$.

with miners competing in a *hash-war*. The task given to the nodes consists of a computational problem that is easily validated by the nodes but at the same time hard to solve since it requires a vast amount of energy.

To evolve from the wasteful PoW consensus to a more energy-efficient Byzantine-fault-tolerant algorithm, PoS was introduced. Using PoS, miners are incentivized to remain honest without the need for a *hash-war*. While PoS encourages decentralization and can improve scalability, the trade-off mostly occurs in terms of security. But as seen in Ouroboros, which has not been evaluated on a large network yet, security is guaranteed by the different protocols implemented around the PoS system assuming liveness and persistence are given. Considering the already successfully implemented PPCoin however, security issues are prevalent, as for example the nothing-at-stake problem. A more secure variant of PoS, DPoS, can be used to further achieve a trade-off between security and scalability versus decentralization.

None of the compared algorithms, except DPoS, could handle a 51% attack. In some cases, a combination of attacks as for example in Nano where a DoS combined with a 51% attack could lead to only needing 33% of the stakes in that respective representative. In case of DPoS, not only 67% of the witnesses would need to be malicious, but also 51% of the voters for a successful attack.

As of now, the trilemma has not been solved by a single consensus algorithm. It is important to evaluate carefully which trade-off to make before choosing a solution.

All in all, law restrictions as well as the efficiency of mining as of this moment lead to the need of more feasible consensus algorithms. Different laws and economical factors may have an influence on the algorithms.

Bibliography

[1] Adam Back, *Hashcash - A Denial of Service Counter-Measure*. August 1, 2002, [On-line] `https://www.researchgate.net/publication/2482110_Hashcash_-_A_Denial_of_Service_Counter-Measure`, last visit May 25, 2019.

[2] Aggelos Kiayias, Alexander Russell, Bernardo David, Roman Oliynykov: *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol*; 37th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 2017), California, USA, August 2017, LNCS, Springer, Cham, Vol. 10401, pp. 357-388.

[3] Andy Johnso, *Weekly Update 4/15/19*. 2019, [On-line] `https://medium.com/nanocurrency/weekly-update-4-15-19-68e6be922699`, last visit May 25, 2019.

[4] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, Srdjan Capkun: *On the Security and Performance of Proof of Work Blockchains*; 23rd ACM Conference on Computer and Communications Security (CCS 2016), Vienna, Austria, October 2016, pp. 3-16.

[5] ash, *Current BitShares Testnet Stress-test Highlight: 3300TXs 14000OPs*. 2017, [On-line] `https://steemit.com/bitshares/@ash/current-bitshares-testnet-stress-test-highlight-3300tx-14000ops`, last visit May 25, 2019.

[6] Ayelet Sapirshtein, Yonatan Sompolinsky, Aviv Zohar: *Optimal Selfish Mining Strategies in Bitcoin*; International Conference on Financial Cryptography and Data Security (FC 2016), Christ Church, Barbados, August 2016, LNCS, Springer, Berlin, Heidelberg, Vol. 9603, pp. 515-532.

[7] Ben Green, *Code cleanup, add test*. January 8, 2018, [On-line] `https://github.com/numtel/node-raiblocks-pow/blob/70d26cde2ab9eed91a168e49136aa46c44a2d052/functions.cpp#L7`, last visit May 25, 2019.

[8] BitFury Group, *Proof of Stake versus Proof of Work*. September 13, 2015, [On-line] `https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/`, last visit April 11, 2019.

[9] BitShares, *BitShares Block Explorer*. n.d., [On-line] `https://wallet.bitshares.org/#/explorer/blocks`, last visit April 16, 2019.

[10] BitShares, *Delegated Proof-of-Stake Consensus*. n.d., [On-line] `https://bitshares.org/technology/delegated-proof-of-stake-consensus/`, last visit April 16, 2019.

[11] BitShares Blockchain Foundation, *The BitShares Blockchain*. June 1, 2018, [On-line] `https://www.bitshares.foundation/articles/2018-06-01-bitshareswhitepaper`, last visit April 16, 2019.

[12] Cami Albert, *Nano (NANO) IOT Charger Sends NANO Coin Price to the Moon*. August 25, 2018, [On-line] `https://cryptoglobalist.com/2018/08/25/nano-nano-iot-charger-sends-nano-coin-price-to-the-moon/`, last visit May 25, 2019.

[13] Cardano, *Introduction*. n.d., [On-line] `https://cardanodocs.com/introduction/`, last visit May 25, 2019.

[14] Cardano, *Ouroboros Proof of Stake Algorithm*. n.d., [On-line] `https://cardanodocs.com/cardano/proof-of-stake/`, last visit May 25, 2019.

[15] Cardano, *cardanoroadmap*. n.d., [On-line] `https://cardanoroadmap.com/`, last visit May 25, 2019.

[16] Cardano, *Why we are building Cardano*. n.d., [On-line] `https://whycardano.com`, last visit May 25, 2019.

[17] Carlo Gutierrez, *Hyperledger's Sawtooth Lake Aims at a Thousand Transactions per Second*. March 13, 2017, [On-line] `https://www.altoros.com/blog/hyperledgers-sawtooth-lake-aims-at-a-thousand-transactions-per-second/`, last visit April 18, 2019.

[18] Christian Cachin, Marko Vukolić: *Blockchain Consensus Protocols in the Wild*; CoRR, July 7, 2017, arXiv:1707.01873.

[19] Colin LeMahieu, *Nano: A Feeless Distributed Cryptocurrency Network*. 2018, [On-line] `https://nano.org/en/whitepaper`, last visit March 3, 2019.

[20] CryptoLions, *EOS Network Monitor*. n.d., [On-line] `https://eosnetworkmonitor.io/#`, last visit April 16, 2019.

[21] Daniel Larimer, *DPOS Consensus Algorithm - The Missing White Paper*. 2017, [On-line] `https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper`, last visit April 18 , 2019.

[22] David Mazières, *The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus*. April 2015, [On-line] `https://www.stellar.org/papers/stellar-consensus-protocol.pdf`, last visit May 25, 2019.

[23] David Z. Morris, *Bitcoin Hits a New Record High, But Stops Short of $20,000*. December 17, 2017, [On-line] `http://fortune.com/2017/12/17/bitcoin-record-high-short-of-20000/`, last visit May 26, 2019.

[24] Digiconomist, *Bitcoin Energy Consumption Index*. n.d., [On-line] `https://digiconomist.net/bitcoin-energy-consumption`, last visit April 11, 2019.

[25] Erik Zhang, *A Byzantine Fault Tolerance Algorithm for Blockchain*. n.d., [On-line] `https://docs.neo.org/en-us/basic/consensus/whitepaper.html`, last visit April 20, 2019.

[26] Fahad Saleh, *Blockchain Without Waste: Proof-of-Stake*. February 27, 2019, [On-line] `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3183935`, last visit May 27, 2019.

[27] Federico Matteo Benčić, Ivana Podnar Žarko: *Distributed ledger technology: blockchain compared to directed acyclic graph*; 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2018, pp. 1569-1570.

[28] Fred B. Schneider: *Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial*; ACM Computing Surveys, Vol. 22, No. 4, December 1990, pp. 299-319.

[29] Ghassan Karame: *On the Security and Scalability of Bitcoin's Blockchain*; 23rd ACM Conference on Computer and Communications Security (CCS 2016), Vienna, Austria, October 2016, pp. 1861-1862.

[30] Giulia Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, Gerui Wang: *Compounding of Wealth in Proof-of-Stake Cryptocurrencies*; CoRR, October 15, 2018, arXiv:1809.07468.

[31] Greg Lee, TestZ, Josh Lavin, Daniel Larimer, Thomas Cox, Nathan Hourt, Qianli Ma, William Prioriello, *EOS.IO Technical White Paper v2*. April 28, 2018, [On-line] `https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md`, last visit April 16, 2019.

[32] Hadja F. Ouattara, Daouda Ahmat, Tounwendyam Frédéric Ouedraogo, Tegawendé F. Bissyandé, Oumarou Sié: *Blockchain Consensus Protocols*; 9th EAI International Conference on e-Infrastructure and e-Services for Developing Countries (AFRICOMM 2017), Lagos, Nigeria, December 2017, LNICST, Springer, Cham, Vol. 250, pp. 304-314.

[33] Hubert Kirrmann, *Fault Tolerant Computing in Industrial Automation*. 2005, [On-line] `http://www.solutil.ch/kirrmann/FaultTolerance/20050418_HK_FT_Tutorial.pdf`, last visit May 26, 2019.

[34] Iddo Bentov, Ariel Gabizon, Alex Mizrahi: *Cryptocurrencies Without Proof of Work*; International Conference on Financial Cryptography and Data Security (FC 2016), Christ Church, Barbados, August 2016, LNCS, Springer, Berlin, Heidelberg, Vol. 9604, pp. 142-157.

[35] Igor M. Coelho, Vitor N. Coelho, Peter Lin, Erik Zhang, *Delegated ByzantineFault Tolerance: Technical details,challenges and perspectives*. March 14, 2019, [On-line] `https://docs.neo.org/en-us/08_dbft.pdf`, last visit May 26, 2019.

[36] Intel Corporation, *Sawtooth v1.1.4 documentation - Introduction*. n.d., [On-line] `https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html`, last visit April 18, 2019.

[37] Intel Software, *Intel Software Guard Extensions*. n.d., [On-line] `https://software.intel.com/en-us/sgx`, last visit April 18, 2019.

[38] Ittay Eyal, Emin Gün Sirer: *Majority is Not Enough: Bitcoin Mining is Vulnerable*; Communications of the ACM, Vol. 61, No. 7, July 2018, pp. 95-102.

[39] James Ray, *Problems*. August 22, 2018, [On-line] `https://github.com/ethereum/wiki/wiki/Problems`, last visit April 15, 2019.

[40] Jean-Claude Laprie: *On Computer System Dependability: faults, errors and failures*; 13th IEEE Computer Society International Conference (COMPCON 1985), California, USA, February 1985, pp. 256-259.

[41] John R. Douceur: *The Sybil Attack*; Peer-to-Peer Systems, First International Workshop (IPTPS 2002), Massachusetts, USA, March 2002, LNCS, Springer, Berlin, Heidelberg, Vol. 2429, pp. 251-260.

[42] Juan Garay, Aggelos Kiayias, Nikos Leonardos: *The Bitcoin Backbone Protocol: Analysis and Applications*; Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015), Sofia, Bulgaria, April 2015, LNCS, Springer, Berlin, Heidelberg, Vol. 9057, pp. 281-310.

[43] Karl J. O'Dwyer, David Malone: *Bitcoin mining and its energy footprint*; 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014), Limerick, Ireland, June 2013, pp. 280-285.

[44] Leslie Lamport, Robert Shostak, Marshall Pease: *The Byzantine Generals Problem*; ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp. 382-401.

[45] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, Weidong Shi: *On Security Analysis of Proof-of-Elapsed-Time (PoET)*; 19th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2017), Massachusetts, USA, November 2017, LNCS, Springer, Cham, Vol. 10616, pp. 282-297.

[46] Loi Luu, Viswesh Narayanan, Kunal Baweja, Chaodong Zheng, Seth Gilbert, Prateek Saxena, *SCP: A Computationally-Scalable Byzantine Consensus Protocol For Blockchains*. December 13, 2015, [On-line] `https://www.weusecoins.com/assets/pdf/library/SCP%20-%20A%20Computationally-Scalable%20Byzantine.pdf`, last visit May 25, 2019.

[47] Mauro Conti, Ankit Gangwal, Michele Todero: *Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages*; CoRR, January 28, 2019, arXiv:1901.10019.

[48] Michael J. Fischer, Nancy A. Lynch, Michael S. Paterson: *Impossibility of Distributed Consensus with One Faulty Process*; Techincal Report No. HIT/LCS/TR-282, September 1982; Department of Computer Science, Yale University, USA.

[49] Miguel Castro, Barbara Liskov: *Practical Byzantine Fault Tolerance*; Third Symposium on Operating Systems Design and Implementation (OSDI 1999), Louisiana, USA, February 1999, USENIX Association, Vol. 99, pp. 173-186.

[50] Nano, *The Incentives to Run a Node*. November 3, 2018, [On-line] `https://medium.com/nanocurrency/the-incentives-to-run-a-node-ccc3510c2562`, last visit May 26, 2019.

[51] Nanode, *Representatives*. n.d., [On-line] `https://www.nanode.co/representatives`, last visit May 26, 2019.

[52] Neo, *Blockchain Info*. n.d., [On-line] `https://neo.org/consensus`, last visit May 27, 2019.

[53] Nichanan Kesonpat, *Consensus Algorithms: Proof-of-Stake & Cryptoeconomics*. June 9, 2018, [On-line] `https://www.nichanank.com/blog/2018/6/4/consensus-algorithms-pos-dpos`, last visit April 16, 2019.

[54] Peercoin, *Comparison with other blockchain networks*. n.d., [On-line] `https://docs.peercoin.net/`, last visit May 27, 2019.

[55] Samuel Eilenberg: *Automata, Languages, and Machines*; Academic Press, Inc., Orlando, Florida, USA, 1974.

[56] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. October 31, 2008, [On-line] `https://bitcoin.org/bitcoin.pdf`, last visit May 26, 2019.

[57] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, George Danezis: *Consensus in the Age of Blockchains*; CoRR, November 14, 2017, arXiv:1711.03936.

[58] Spencer J. Hosack: *Use of the Proof-of-Stake Algorithm for Distributed Consensus in Blockchain Protocol for Cryptocurrency*; Honors Scholar Theses, University of Connecticut, USA, Supervisors: Yaacov Kopeliovich, 2018.

[59] Stellar Development Foundation, *On Worldwide Consensus*. April 8, 2015, [On-line] `https://medium.com/stellar-development-foundation/on-worldwide-consensus-359e9eb3e949`, last visit April, 18, 2019.

[60] Steve Walters, *Delegated Proof of Stake (DPoS): What is It? - Complete Beginners Guide*. August 20, 2018, [On-line] `https://www.coinbureau.com/education/delegated-proof-stake-dpos/`, last visit April 16, 2019.

[61] Stuart Haber, W. Scott Stornetta: *How to Time-Stamp a Digital Document*; 10th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 1990), California, USA, August 1990, LNCS, Springer, Berlin, Heidelberg, Vol. 537, pp. 437-455.

[62] Sunny King, Scott Nadal, *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. August 19, 2012, [On-line] `https://decred.org/research/king2012.pdf`, last visit May 25, 2019.

[63] Thomas Bocek, Burkhard Stiller: *Smart Contracts - Blockchains in the Wings*; in Claudia Linnhoff-Popien, Ralf Schneider, Michael Zaddach (Edts.) "Digital Marketplaces Unleashed", Springer, Berlin, Germany, 2018, pp. 169-184.

[64] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, Ji Wang: *Untangling Blockchain: A Data Processing View of Blockchain Systems*; IEEE Transactions on Knowledge and Data Engineering, Vol. 30, No. 7, July 2018, pp. 1366-1385.

[65] Tyler Jenks, *Pros and Cons of the Delegated Proof-of-Stake Consensus Model*. August 16, 2018, [On-line] `https://www.verypossible.com/blog/pros-and-cons-of-the-delegated-proof-of-stake-consensus-model`, last visit April 18, 2019.

[66] Vitalik Buterin, *Long-Range Attacks: The Serious Problem With Adaptive Proof of Work*. May 15, 2014, [On-line] `https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/`, last visit April 14, 2019.

[67] Vitalik Buterin, *Proof of Stake FAQ - How does validator selection work, and what is stake grinding?*. March 20, 2019, [On-line] `https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ#how-does-validator-selection-work-and-what-is-stake-grinding`, last visit April 14, 2019.

[68] Vitalik Buterin, *Proof of Stake FAQ - What is the "nothing at stake" problem and how can it be fixed?*. March 20, 2019, [On-line] `https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ#what-is-the-nothing-at-stake-problem-and-how-can-it-be-fixed`, last visit April 14, 2019.

[69] Vitalik Buterin, *Sharding FAQ*. March 20, 2019, [On-line] `https://github.com/ethereum/wiki/wiki/Sharding-FAQ#this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it`, last visit April 15, 2019.

[70] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, Dong In Kim: *A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks*; IEEE Access, Vol. 7, January 2019, pp. 22328-22370.