University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Proof of Stake for Bazo

Simon Bachmann
Zurich, Switzerland
Student ID: 14-709-893

Supervisor: Thomas Bocek
Date of Submission: February 1, 2018

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Proof-of-Stake (PoS) ist eine Klasse der Konsensus-Algorithmen, die in Bezug auf Stromverbrauch und Sicherheit signifikante Verbesserungen gegenüber dem derzeit marktbeherrschend Proof-of-Work (PoW) Konsensus-Mechanismus bietet.

Der PoS-Konsensus-Algorithmus wird im Allgemeinen am besten wie folgt beschrieben: Ein Validierer ist ein Knoten im Netzwerk, der Transaktionen verifiziert und diese in From eines Blocks der Blockchain hinzufügt. Jeder Knoten in dem System kann zu einem Validierer werden, indem eine bestimmte Anzahl Tokens der dazugehörigen Kryptowährung hinterlegt wird. Diese Tokens können nicht ausgegeben werden, solange der Validierer Teil des Validierersets ist. Diese Kaution wird verwendet, um bösartige Knoten zu bestrafen. Das Recht für das Hinzufügen des nächsten Blocks wird in einem dezentralisierten Auswahlverfahren gemacht und ist abhängig von der Anzahl der hinterlegten Tokens jedes Validierers.

Diese Bachelorarbeit zeigt die Vorteile von PoS-Protokollen gegenüber einem PoW-Konsensus-Mechanismus auf. Dabei wird auf den Stromverbrauch von Bitcoin eingegangen, der zur Zeit des Verfassens dieser Arbeit auf mehr als 5 Millionen USD pro Tag geschätzt wird. Ebenfalls wird auf das Risiko der Zentralisierung von Mining-Pools und Cloud-Mining-Platformen eingegangen und wieso die Eintrittsbarriere ins PoS-Validieren tiefer als in PoW-Mining ist.

Weiter werden die Unterschiede zwischen den bereits existierenden PoS-Konsensus-Algorithmen aufgezeigt und analysiert. Aus diesen Analysen werden die allgemeinen Schwierigkeiten in PoS-Systemen dargelegt.

Bazo ist eine Kryptowährung basierend auf der Blockchain Technologie, die in einer früheren Bachelorarbeit an der Universität Zürich entwickelt wurde. Es erlaubt die Erstellung von neuen Benutzerkonten und das Transferieren von Bazo Tokens zwischen Benutzerkonten. Aus Einfachheitsgründen hatte man sich damals für einen PoW-Konsensus-Mechanismus entschieden. Das Ziel dieser Bachelorarbeit ist es, das Konsensus-Protokoll der Bazo Blockchain in ein PoS-System zu überführen.

In a previous bachelor thesis of the University of Zurich the Bazo cryptocurrency was developed. It supports the creation of new accounts and the transfer of funds on the blockchain. Due to simplicity Bazo uses a Proof-of-Work (PoW) consensus algorithm. The biggest disadvantages of a PoW consensus mechanism are the high energy consumption, the risk of centralization and the high entry barrier for new miners. Furthermore, in early stages of a new cryptocurrency a blockchain with a low number of miners poses a great

risk of the 51% attack. A successful implementation of a PoS-protocol could solve these problems mentioned above.

The goal of this bachelor thesis is to study and analyse current Proof-of-Stake (PoS) implementations, design a new system based on the findings and convert the Bazo consensus mechanism from PoW to PoS.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Thomas Bocek for the continuous support of my Bachelor's Thesis. His patience, motivation and profound knowledge in the field of blockchain technology is inspiring.

Besides my supervisors I would also like to thank Prof. Dr. Burkhard Stiller, the head of the Communication Systems Research Group, for giving me the opportunity to work on such an interesting and currently relevant topic.

Contents

Chapter 1

Introduction

Proof-of-Stake (PoS) is a class of consensus algorithm which provides significant improvements in terms of electricity consumption and security towards the currently most dominant consensus mechanism called Proof-of-Work (PoW).

A PoS consensus algorithm is in general best described as follows: A validator is a node in the network that validates transactions and adds them to the blockchain. Any node in the system can become a validator by depositing tokens of the associated cryptocurrency. These tokens cannot be spent as long as the validator is part of the validator set. This deposit is used to punish malicious nodes. Validators are elected by the network proportionally to the deposited funds for appending the next block. The right to add the next block is made in a decentralized selection procedure depending on the number of tokens deposited by each validator.

The category of PoS consensus mechanisms consists of many different algorithms and protocols, which each solves the tasks of random validator election, reward system and other PoS challenges differently. There are two major types of algorithms for PoS protocols: chain-based and Byzantine agreement protocols, which will be discussed in more detail in Section 2.1.

1.1 Motivation

PoS addresses several risks and efficiency improvements over PoW systems which are discussed in this Section.

1.1.1 Environmental Harm

By the time of writing this thesis, Digiconomist [1] reports that the Bitcoin blockchain consumes more than 36 TWh annually. For comparison, this is more than the entire nation of Bulgaria consumes every year and they are ranked 60th position in the global energy

consumption rating. In other terms the Bitcoin blockchain uses as much electricity as more than 3 million households together. Thus, in order to secure the Bitcoin blockchain the global mining costs amount to over 5 million USD every day.

There is no need for an extensive computation task in a distributed system that uses PoS as a consensus mechanism in order to have a similar level of security. With new emerging blockchains almost every day, an environmentally friendly and sustainable consensus mechanism that replaces PoW is needed.

1.1.2 Risk of Centralization in Form of Mining Pools

The chances of adding a block for an individual minier in PoW is extremely low. Therefore, miners can join so-called mining pools where their computational power is collaboratively deployed. The block reward and transaction fees are split among the members of the mining pool proportionally to the provided computational effort. This guarantees a constant revenue stream.

By the time of writing this thesis, the largest mining pools (BTC.com, BTC.TOP, AntPool) represent more than 50% of the total hashing power [3]. If these mining pools joined together, the Bitcoin blockchain would be vulnerable to a 51% attack where this mining pool could always provide a longer competing chain than the currently accepted one [4]. Therefore, the Bitcoin blockchain would be controlled by a single centralized entity.

The need for a constant revenue stream is not required in a PoS system since the validator does not have to be compensated for burned resources. Therefore, the formation of large validator sets is less likely.

1.1.3 Risk of Centralization in Form of Cloud Mining

Cloud-mining allows people to mine cryptocurrencies without owning mining hardware [5]. Companies providing cloud-mining services profit from economies of scale such as special deals on hardware orders and lower maintenance costs. Therefore, these companies gain a economical advantage over individual miners and may force them out of the market.

Since no specialized hardware is needed for a PoS system, centralized cloud companies do not provide any significant advantage.

1.1.4 Entry Barrier

A miner in a PoW system with specialized hardware gains an edge over the competition, whereas in a PoS system a validator can only generate revenue proportionally to his/her deposits in the system. The entry barrier of becoming a validator in a PoS system is therefore significantly lower than becoming a miner in a PoW system.

1.1.5 Discrepancy between Miners and Non-Miners

Another advantage of PoS protocols is that the discrepancy between mining and non-mining nodes can be considerably lowered. Practically any machine with enough storage and bandwidth can be part of the validator set. Therefore, the community is less split up as it occurs in PoW implementations.

1.1.6 51% Attack

In the initial phase of a new blockchain the number of validators/miners is limited. This poses a great risk of a 51% attack. A malicious user can easily buy mining hardware such that he/she possesses at least 51% of the total hashing power of the network. This user can produce a longer competing chain than the rest of the network. The system always follows the longest competing chain and therefore, the malicious user has control of the blockchain.

The developers of a PoS system can hold back 51% of the coins until the network is established in order to prevent such an attack by an outsider.

1.1.7 Transaction Fees

Transaction fees prevent a blockchain from being spammed but also compensate the miners for their computational effort and electricity costs. No resources are burned in a PoS consensus mechanism and therefore, the transaction fees can be significantly lowered.

1.2 Description of Work

This thesis covers the design of a PoS mechanism for the Bazo cryptocurrency that allows to reduce the energy consumption for mining. Furthermore, it increases security as mining without many miners poses a great risk of a 51% attack. Currently, the Bazo cryptocurrency uses a SHA3 partial hash collision as a Proof-of-Work (PoW) consensus mechanism. The main reason to use this in a first implementation is its simplicity, because a SHA3 partial hash collision requires only a few lines of code. Since it is difficult to implement a fully decentralized PoS, delegated PoS (DPoS) or a PoS/PoW scheme have to be considered.

The work of this Bachelor's Thesis is highly explorative, technical, and prototype-driven in finding the right optimizations for the PoS mechanism. At the same time, this requires at the same time a clear, detailed, and technically sound documentation of optimization steps performed, including the protocol design itself, a detailed reasoning of those steps selected, and outlining the scientific value of those mechanisms under study.

1.3 Thesis Outline

This thesis is structured in five Sections. Section 1 focuses on the motivation of a PoS protocol and its advantages. Chapter 2 familiarizes the reader with current PoS implementations and explains the difficulties behind such a protocol. Section 3 concentrates with the design and the implementation of the new consensus mechanism for the Bazo cryptocurrency, based on the given requirements. An evaluation of the PoS consensus mechanism is conducted in Chapter 4, which is followed by Chapter 5 about future work. A summary and conclusions are drawn in Chapter 6.

Chapter 2

Related Work

This chapter introduces the two different types of PoS consensus mechanisms that currently exist. Further, it explains what challenges PoS algorithms face in general and what possible attacks must be prevented when designing a new consensus mechanism.

2.1 Different Types of PoS

From all the currently available implementations of PoS consensus mechanisms one can distinguish between two categories - chain-based protocols and Byzantine agreement protocols. Both types are described thouroughly by means of successful implementations in other blockchains.

Both types consider the amount of coins that each validator has deposited in the system in order to create a payout proportionally to the number of coins that are possessed by a validator. A validator with 50 coins should on average receive five times the staking reward that a validator with only 10 coins does.

2.1.1 Chain-Based Protocols

In this category of PoS consensus algorithms a validator is chosen randomly from the validator set in order to append the next block. This random election solely depends on information that is stored within the blockchain, such as the height of a particular block or the amount of coins a validator possesses.

Peercoin (PPCoin)

In PPCoin the right of appending the next block is assigned to the validator that is in compliance with the condition (2.1) [10].

$$\frac{hash(PrevBlockData \cdot TimeInSec \cdot TxOut)}{Coins(TxOut) * TimeWeight(TxOut)} \leq Target \tag{2.1}$$

TxOut A validator must deposit a minimum amount of coins to an unspendable address when joining the set of validators. *TxOut* represents this transaction, which is different for every individual validator.

Target This constant determines the speed of the blockchain.

Coins(TxOutA) This parameter represents the amount of coins that a validator has deposited into the unspendable address.

TimeWeight(TxOut) PPCoin uses a mechanism called coin aging [11] which prevents stakeholders with a significant fraction of all the coins from adding multiple consecutive blocks. This variable is reset when a validator has successfully appended a block. Coin aging makes the protocol vulnerable to attacks. A malicious node can hold back coins in multiple addresses for a long period of time to accumulate a large coin age. The likelihood of being elected multiple times in a row is increased proportionally to the accumulated coin age. Due to this reason PPCoin limited the maximal amount of *TimeWeight(TxOut)* to 90 days in their v0.3 protocol [10].

TimeInSec PPCoin's random election process is similar to the algorithm that is used in PoW with one important difference. The only variable parameter is *TimeInSec* (validators local clock) and therefore slows down the hash rate to one attempt per second no matter what hardware is being used.

PPCoin is vulnerable to stake grinding attacks where a validator can influence parameters in order to have a higher probability to be reelected for the next block. This is because the data of the previous block is included in the PoS condition. [7]

2.1.2 Byzantine Agreement Protocols

In Byzantine agreement protocols validators agree upon the next block by means of a multi-round voting process. Validators are randomly selected in order to propose the next block. Each validator has a voting power according to their stake in the system. In each round validators can commit to a block. If they agree to the proposed block and if the block receives a majority of votes, it is considered as valid. All messages that are used for proposing a new block or committing to a block are done handled by the gossip protocol. This means none of these messages are recorded on the blockchain.

Tendermint

The state machine shown in Figure 2.1 illustrates how the round based consensus protocol of Tendermint [6] works. A validator contributes to the consensus by signing votes for

proposed blocks. Each round is split up in three steps: *Propose*, *Prevote* and *Precommit*, followed by two special steps called *Commit* and *NewHeight*.

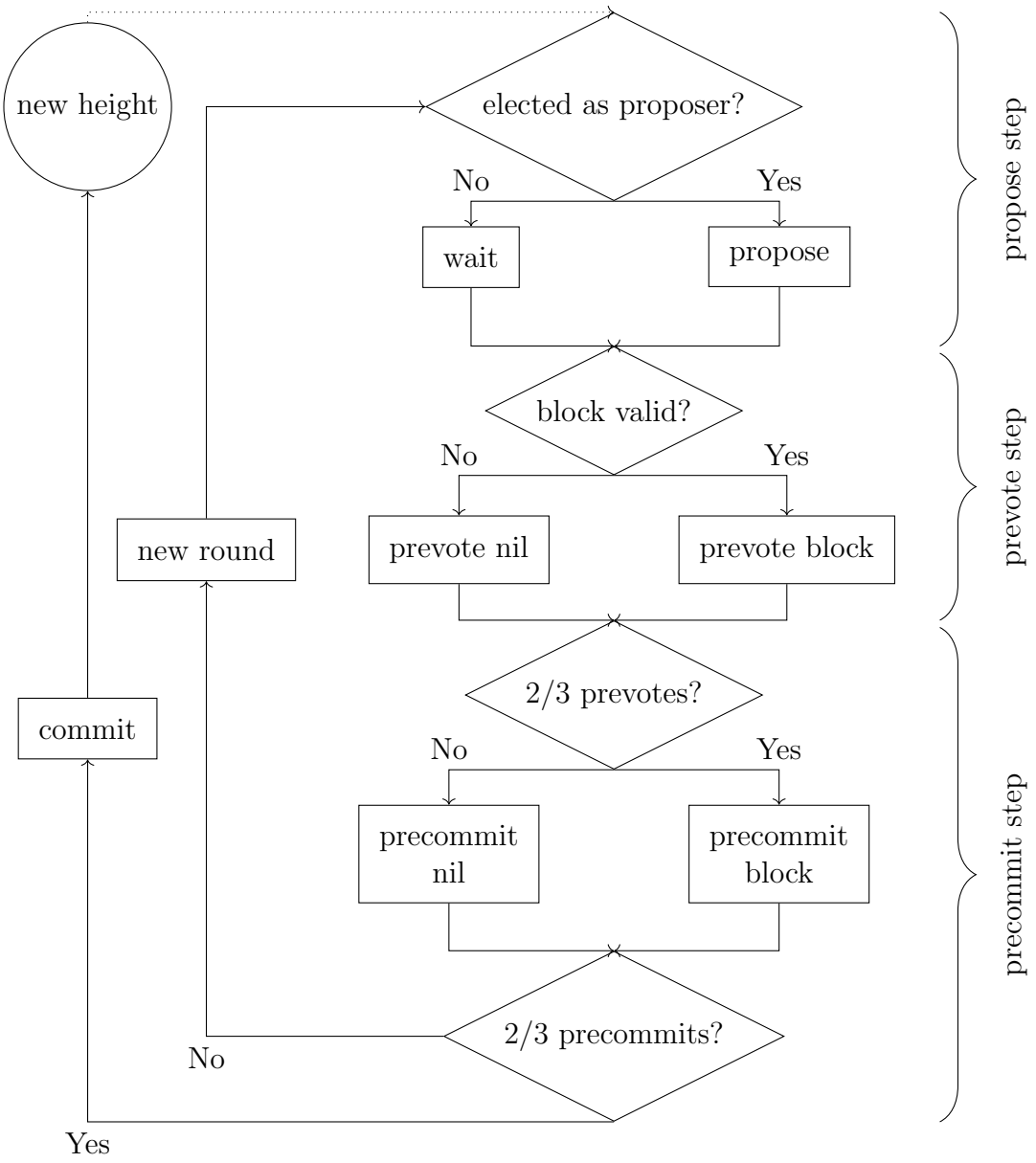

Figure 2.1: Tendermint State Machine [8]

Propose In every round a validator is nominated as a block proposer in round-robin fashion. The frequency of being selected depends on the validators stake. The higher the stake the higher the chances of being elected. The elected validator creates a block and broadcasts it as a proposal. All other validators listen for the broadcasted block.

Prevote Every validator has to make a decision during this step. A validator can accept the block by broadcasting a signed *Prevote* message. If a node has not received any proposal or an invalid block, it signs and broadcast a *Nil Prevote*.

Precommit In this step each node validates if it received more than $\frac{2}{3}$ of the total *Prevotes*. If this is the case the validator signs and broadcasts a *Precommit* message. At the end of this step a note decides again if it received more than $\frac{2}{3}$ of *Precommits*. Depending on this decision the validator either enters the *Commit* step or starts a new round in the *Propose* step with a slightly increased time period for each step.

Commit A validator signs and broadcasts a *Commit* message. As soon as a node has received the block and at least $\frac{2}{3}$ of the *Commits* it considers the block final and enters the *NewHeight* step.

NewHeight The purpose of this time window is to gather additional commits for the previously committed block.

One might think that the *Precommit* step is unnecessary but there are some more mechanism needed [8] that take place within that step, which are not discussed within this paper.

Algorand

Algorand's PoS protocol is similar to the protocol of Tendermint with significant improvements.

In Tendermint the right of proposing the next block is being passed in round-robin fashion [6]. Algorand uses a mechanism based on Verifiable Random Functions that allows nodes to privately check whether they are part of the current round. After gossiping a message to the network these nodes are immediately replaced. This prevents DoS attacks on chosen validators after their identity is revealed.

Algorand addresses the scalability problem that exists in other protocols. It achieves scalability by randomly selecting a committee which is a small subgroup of representatives of all validators. Only the committee is eligible to run a certain step in the protocol which reduces the number of messages passed through the network. Algorand claims that the protocol is able to handle 125 times more transactions than Bitcoin does [9].

2.2 Challenges

There arise new difficulties and possible attacks with PoS consensus mechanisms. This section explains what mechanisms are needed to prevent nodes from double spending. Other difficulties such as the manipulation of the random election, DoS attacks as well as the difficulty from keeping validator online at all time are outlined.

2.2.1 Nothing at Stake (Double Spend)

In the case where a blockchain forks and multiple chains are competing with each other, a validator has two or more options where to add the next block. Unlike in PoW, there are no resources burned in the process of finding a random node in the network. This allows the validator to append a block on any of the competing chain. Therefore, it is in the validators incentive to build on top of every competing chain. This strategy assures that the suggested block will most likely be included in the chain which will be accepted by the network.

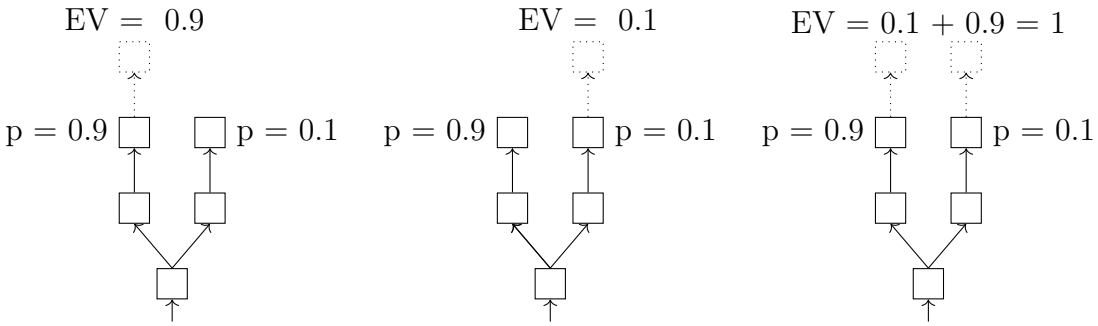

Figure 2.2: Competing Chains in PoS [7]

If all validators act economically and append their block on every competing chain, the blockchain will never reach consensus, even if there are only honest validators. Figure 2.2 shows how the expected value EV changes depending on the validators strategy.

The value of p depends on the percentage of validating nodes that received this chain prior to any other chain of the same length. The fractional amount of the total stake that votes for a particular chain influences p as well. For simplicity it is assumed that the block-reward and transaction fees add up to exactly one coin for each block.

Unlike in PoW, where a mining node would need to split its hashing power in order to vote on multiple chains. This is due to the reason that the hash of the previous block is included in the PoW calculation. Therefore, a miner will always put all his mining power into the longest chain, which is the chain that will most likely be accepted by the network (see Figure 2.3).

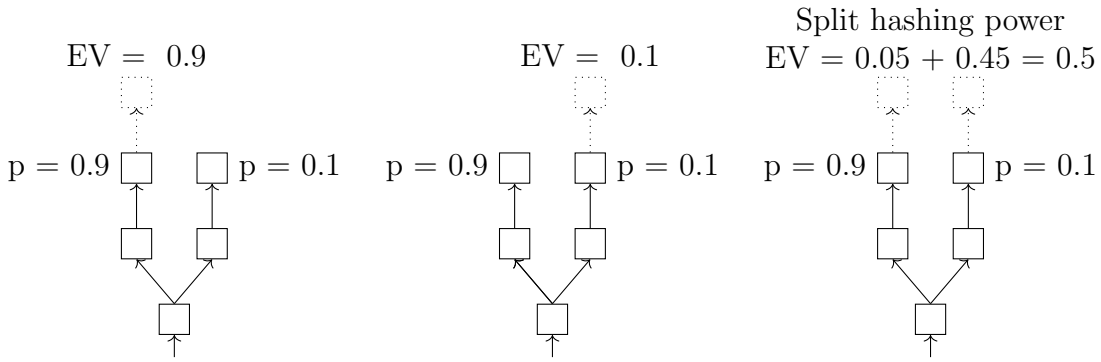

Figure 2.3: Competing Chains in PoW [7]

In the context of PoW, the probability p of a chain depends on the percentage of mining nodes that received this chain prior to the any other chain of the same length. The fraction of the total hashing power that is put into each chain influences the likelihood p aswell. Consequently, miners have to make a decision on what competing chain they want to continue if they have two or more competing chains of the same length. This results in a separation of the miners and therefore, p has to add up to exactly 100 percent over all the competing chains of all the miners.

As previously described, rational validators in a PoS system append a block on every competing chain. Therefore, the sum of the amount of voting power from all competing chains can be more than 100 percent. This property is described in the following scenario and shown in Figure 2.4. After block a is added, two validators F and G append a block at the same time which results in a fork with two competing chains. Every validator wants to make sure that his/her block will be included in the finalized blockchain. Therefore, every following block is added on both chains.

With the assumption that validator F has a staking power of 1% and validator G of 2% and all the validators between block f and y make up for the total stake of 96%. At the time of block y the upper chain has a staking weight of 98% and the lower chain of 97%. The sum of the amount of voting power of both competing chains in this example is 1.96 which is clearly higher than 1.

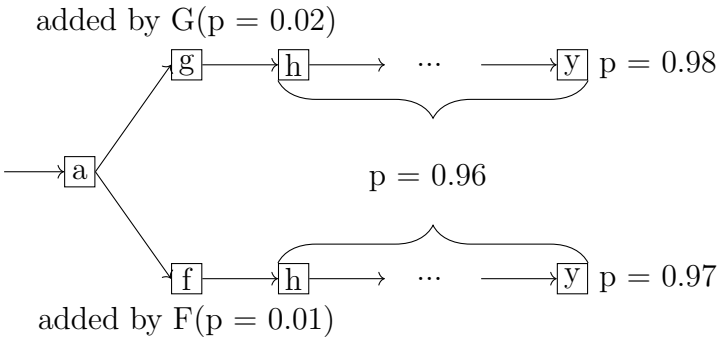

Figure 2.4: Staking Weights in PoS [7]

A malicious user Z can take advantage of such a situation and double spend his coins. A transaction that is included in block g but not in f can easily be reversed by appending a new block z on only the subchain where block g is not included. Therefore, after user Z added block z, the chain containing block f and z has a higher staking weight and will be accepted by the network as shown in Figure 2.5. In this example the malicious user has double spent his coins with only 2% of the total stake.

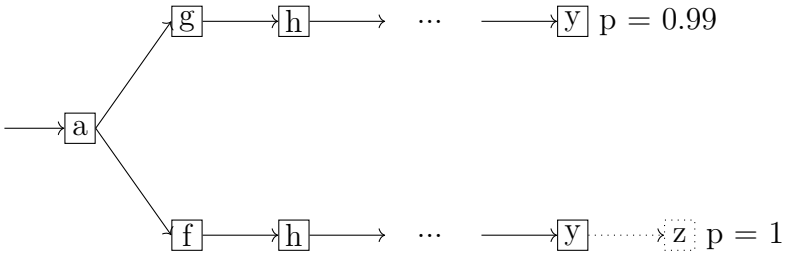

Figure 2.5: Double Spending in PoS [7]

To ensure a secure implementation of PoS, a protocol has to implemented that allows validators to be punished for appending new blocks on multiple chains at the same height.

2.2.2 Grinding Attack

When validators are able to manipulate the random election process of the consensus algorithm, it creates the possibility of grinding attacks.

For example a poor design is when the election process depends on the previous block-hash. The node which is elected for adding a block can manipulate the block-hash by in- or excluding certain transactions which then results in different block-hashes. Hence, this node can grind through many different combinations and choose the one that reelects itself with a high likelihood for the next block.

Therefore, the protocol must make sure that the parameters for the random election process cannot be modified at commitment time. This can be achieved by committing to a piece of information far in advance and is described in more detail in Section 3.1.1.

2.2.3 DoS Attack

If the random election process is determined in a public manner, the elected validator is known ahead of time and the protocol is vulnerable to DoS attacks. Therefore, it is advantageous to hold elecetion process privately.

In a PoW protocol, this is done in a way that each validator tries to find a solution to a mathematical puzzle. This puzzle is different for every miner. A malicious user cannot predict which node solves this puzzle first. Thus, the elected node cannot become a victim of a DoS attack unless the malicious user attacks every node in the network.

A similar level of unpredictability is desired for a PoS protocol.

2.2.4 Availability

Many different parties competing for the next block decreases the possibility of appending multiple consecutive blocks by a single validator. Hence, the network becomes more secure with every additional validator. There are PoS protocols, where the probability of adding the next block increases proportionally to the time that a validator has not been elected. Often this is referred to as coin aging and is described by the means of the PPCoin implementation in Section 2.1.1.

A protocol that includes coin age incentivizes validators for not being online until they reach a high or even the maximum possible coin age. Coin age is accumulated even if a node is offline. In such a scenario, a large fraction of the validator set would then be offline most of the time. Therefore, the actual number of competing nodes in the network is by no means the size of the validator set. A system becomes only more secure with additional validators if these nodes are online and compete for the next block at any time.

A PoS protocol must incentivize validators to be online at any given time.

Chapter 3

Design and Implementation

This chapter covers the detailed description of the protocol design and how the existing Bazo blockchain is revised and extended.

A chain-based protocol was developed for simplicity reasons. This type of protocol works similar to a PoW consensus mechanism with some key differences. It can also be referred to a throttled PoW algorithm since a validator is limited to exactly 1h/s (hash per second) no matter how powerful his/her machine is. It is a semi-synchronous system, which means that every node has its own local time running. If a node attempts to speed up his hashing power, the system detects the malicious node and the suggested blocks are rejected.

The protocol also chooses validators proportionally to the number of coins that each validator owns. Furthermore, it uses immutable random strings (seeds) at every block height in order to determine the next eligible validator. The detailed explanation of the election process is found in Section 3.4.

One of the advantages of the newly designed protocol is that a validator is always elected unless all validators are offline at the same time. Other PoS-protocols must implement a fall-back mechanism that deals with a deadlock, which occurs if all members of a chosen subset of validators are offline.

3.1 Transactions

Transactions are used to change the state of the accounts in the network. There are three transaction types already existing in the network (account creation, transferring funds and adjusting system parameters [12]). This section covers how the transaction types of changing system parameters (Section 3.1.2) and transferring funds (Section 3.1.3) are updated as well as a new type of transaction for joining/leaving the validator set (Section 3.1.1) is introduced.

3.1.1 Stake Transaction (StakeTx)

A node in the active validator set confirms transactions proportionally to their stake in form of blocks of transactions. This new type of transaction allows a node to join or leave the validator set.

The type of transaction consists of the following parameters:

Fee The fee payment is an incentive for the validator to include the transaction in the next block. The higher the amount of the fee, the more likely the transaction will be included in the next block. As in a traditional PoW based protocol, the validator that appends the next block will be rewarded with the transaction fees of all the transactions within the appended block. The amount of the fees can be drastically lowered compared to a PoW system since a validator does not have to be compensated for electricity costs. A small amount still has to be paid for every transaction in order to prevent to blockchain from being spammed.

Is Staking This boolean states whether the node wants to join or leave the set of validators.

Account The hash of the public key of the issuer.

Hashed Seed When a node wants to join the set of validators it must create a seed which is a random 32 bytes long string. By submitting the hash of this seed, the node commits to this seed without actually revealing it and it cannot change the seed in a later step. The same seed will be used in the PoS condition which is described in more detail in Section 3.4.

Signature The signature serves the purpose of authentication. The node digitally signs the transaction with its private key.

A transaction of this type is accepted by the network if the issuer fulfills the minimum staking amount that is needed to become a validator. This minimum amount is a system parameter and described in more detail in Section 3.1.2. Staking transactions are referred to as StakeTx for the rest of the thesis.

3.1.2 System Parameters (ConfigTx)

The idea behind this type of transaction is that the system parameters can be adjusted without the need of a hard fork. The transaction must be signed by a root account in order to be accepted. The existing configuration transaction is extended by the following parameters.

Minimum Staking Amount This is the minimum amount of coins that a potential validator must possess. A node cannot join the set of validators unless it fulfills this requirement. As long as a node is part of the validator set its balance must never fall below the minimum.

Minimum Waiting Time This is the minimum number of blocks that a validator must initially wait for when joining the validator set. After appending a block to the blockchain a validator must also wait the same number of blocks before another one can be added. The reason for waiting a certain nuber of blocks is the prevention of a stake grinding attack. How such an attack would look like is described in Section 4.1.2.

Accepted Time Difference An important characteristic of a semi-synchronous system is the different clock speed in every node in the network. This parameter defines the acceptable time difference. A malicious validator that speeds up his/her clock interval should not be able to append a block to the blockchain.

Slashing Window Size As described in Section 2.2.1 validators must be punished when adding blocks on two competing chains. Otherwise the network will never reach consensus and it offers the possibility for double spending attacks. The slashing window size forces validators to commit to one of the competing chains after a fork. If a validator votes on two different chains within a span of a certain block height he/she will be punished.

Slashing Reward This parameter refers to the amount of coins that a validator receives for providing a correct slashing proof. This incentivizes validators to check if other validators have built on top of multiple competing chains within the slashing window.

3.1.3 Funds Transaction (FundsTx)

This type of transaction is used to transfer funds from one account to another. The validation process of this transaction is slightly modified. The minimum account balance that an active validator must hold is increased to the corresponding system parameter. Therefore, any transaction from an active validator node will be rejected if the balance of the node is below the minimum staking amount after the transaction.

3.2 Account

The Bazo blockchain is an account-based model which means that the union of all the accounts make up the state of the network. Besides the three existing parameters - address, balance and and transaction count - three additional parameters are introduced:

Is Staking This boolean describes whether the account is currently part of the active validator set or not.

Hashed Seed The hashed seed is initially set to 0. As soon as a node joins the set of validators it must commit to a seed by publishing the hash of the seed. This parameter is set as soon as the corresponding account has successfully submitted a StakeTx to the blockchain.

Staking Block Height The system registers the block height at which an account joined the set of validators. This parameter is needed for the slashing condition which is described in more detail in Section 3.5.

3.3 Block

The following parameters are updated from the existing protocol or added to the block parameters:

Number of StakeTx Due to the newly introduced StakeTx this parameter corresponds to the number of StakeTxs that are included in the block.

StakeTx Data The hashes of all StakeTxs that are included in this block in sequential order.

Time in Seconds (updated Nonce) The meaning of the nonce in the updated protocol bears the number of seconds that a validator needs in order to fulfill the PoS condition (Section 3.4).

Seed The validator must publish the seed that it previously committed to when joining the set of validators.

HashedSeed This parameter stores the most recently submitted hash of the seed which a validator has committed to. This parameter is set when a validator joins the validator set (StakeTx). It is also updated after a node successfully added a new block. To stay part of the validator set the node creates a new seed and publishes the hash of the seed. This is necessary since the previously submitted seed will be published as part of newly added block and therefore, the PoS election for this node would no longer occur privately.

Height The height of a block refers to the number of previously appended blocks to the blockchain. This parameter is needed for the PoS condition (Section 3.4) as well as the slashing condition (Section 3.5).

SlashedAddress A validator can submit a slashing proof when appending a block. Therefore, a validator checks if another validator has built on two competing chains within the block span of the slashing window size. This parameter is set to 0 by default if no proof is included. Otherwise, it holds the address of the misbehaving node that must be punished.

Two Conflicting Block Hashes These two parameters exhibit the block hashes where the same node has appended a block on two competing chains within the slashing window size.

3.4 PoS Condition

The PoS condition works similar to the PoW condition but with different parameters and a regulated hash rate for each validator. Each parameter has its own unique function in order to secure the system.

After a node has joined the set of validators and the minimum waiting time (see Section 3.1.2) has passed, it is eligible to append blocks to the blockchain. For doing so it must provide a valid *TimeInSeconds* for the PoS condition 3.1.

$$\frac{SHA\text{-}256([Seed_{PrevBlocks}] \cdot Seed_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target \quad (3.1)$$

Each of the parameter provides an important property:

List of the Previous Seeds($Seed_{PrevBlocks}$) An important characteristic of the seed which is included in every block is the immutability of the seed. A validator has previously committed to this seed and cannot change it during commitment time. It is called the commitment time when a node adds transactions in form of a block to the blockchain by broadcasting the proposed block. The seed is used for the random election process for the upcoming block. As highlighted in Section 2.1.1 a node must not be able to modify the random election process during commitment time. By including a list of the previous seeds a stake grinding attack becomes unfeasible. In Section 4.1.2 stake grinding attacks are revisited and evaluated.

Local Seed($Seed_{Local}$) All parameters in the Condition 3.1 are the same for each validator in the network except the local seed. Without the local seed parameter, the node with most coins would always meet the condition first (smallest $TimeInSeconds$). The local seed individualizes the PoS condition to each validator and therefore, a validator with a low amount of coins also has the possibility to append a block to the blockchain.

Height of the Block($BlockHeight$) The height of a block is characterized by the number of previously added blocks in the blockchain plus one.

Amount of Coins($Coins$) By dividing up the number of coins that a validator possesses, the election process becomes proportional to the stake. Without this division every economically acting node would create new accounts that possess exactly the minimum staking in order to maximize its staking reward.

Difficulty of the PoS Condition($Target$) Similar to the PoW condition [12] the difficulty in the PoS protocol can be adjusted with this global variable in order to determine the speed of the blockchain. As the number of validators in-/decreases the difficulty is adjusted accordingly. The $Target$ is recalculated and adjusted based on the measured average block interval [12].

3.5 Validation

When a new block is received each validator checks the following conditions whether the block is valid or not.

Minimum Waiting Time The minimum waiting time requirement is needed in order to prevent stake grinding attacks. Such an attack is described in Section 4.1.2. The larger the number of blocks the more difficult it becomes for an attacker to perform such an attack. Furthermore, it also sets a validator offline for the number of blocks that is set as the minimum waiting time after appending a block. This is because when adding a block, a new seed is submitted and this also opens the possibility of a stake grinding attack. Therefore, it is desired to adjust the size of the minimum waiting time according to the size of the validator set. The size of the minimum waiting time can be adjusted with a ConfigTx signed by a root account.

Hashed Seed In order to be eligible to append blocks to the blockchain, a node must have previously submitted a hashed seed to join the set of validators. This hashed seed can be submitted as part of a StakeTx or as part of a previously submitted block. This seed is stored in each validators account. Each validator can reconstruct the hash of the seed since the seed is included in the proposed block. If the hashes do not match, the validator has changed the seed during commitment time. In that case the block is rejected.

PoS Condition The current state of the system and the suggested block contain all the needed information to check whether the PoS condition is valid or not.

Slashing Condition When the suggested block includes a proof for a slashing condition, each validator checks for the proof's validity by comparing the height of the two blocks and whether they arise in competing chains or not. If the proof is valid, the beneficiary address receives an addition reward which is determined by a system parameter. Further, the slashed address loses its position in the validator set as well as the minimum amount of coins that is required to be part of the validator set.

Clock Speed When a node tries to manipulate its clock speed, the suggested block will be ignored if the submitted time is too far in the future. This threshold is set with a system parameter.

3.5.1 Competing Chains

Due to latency and the nature of the PoS condition it is possible that blockchain forks and two or more chains are competing each other. In such a case the protocol follows the same principle as in a PoW system:

1. **Same Height**: A validator takes the first received valid block and rejects all blocks belonging to a chain that are of the same height or shorter.

2. **Longer Chain**: If a block is received that belongs to a longer valid chain, this chain is used as the valid chain and a rollback is necessary [12].

Chapter 4

Evaluation

Security considerations directly influence the design of the new consensus mechanism and a range of attacks are mitigated by the employed counteractions. This chapter focuses on the vulnerability of the newly designed protocol as well as it evaluates the efficiency compared to PoW algorithms.

4.1 Attacks

This chapter focuses on the attack scenarios and installed countermeasures in order to mitigate them. It elaborates on Denial-of-Service (DoS) and stake grinding attacks.

4.1.1 DoS Attack

The designed protocol offers the same degree of security towards DoS attacks. The election process is done in the same private manner as in PoW consensus algorithms. After a node has successfully added a new block and therefore revealed its private seed, a new seed is created and the hash of this seed is included in the new block. An attacker cannot predict what the value of the new seed is. Therefore, the implemented system is not more vulnerable towards DoS attacks than it was with a PoW algorithm.

4.1.2 Stake Grinding Attacks

This section evaluates how a stake grinding attack becomes infeasible due to the properties of the PoS condition from Section 3.4. It elaborates on the parameters that are used in the PoS condition and highlights their importance to prevent stake grinding attacks.

Condition 4.1 could be a potential PoS condition.

$$\frac{SHA\text{-}256(Seed_{PrevBlock} \cdot Seed_{Local} \cdot TimeInSeconds)}{Coins} \leq Target \tag{4.1}$$

However, a vicious user could take advantage of the security flaw of this condition and perform a stake grinding attack. Such an attack can occur as follows: This malicious user controls two addresses A and B. Both addresses must be in possession of the minimum amount of coins that is required to join the validator set. For both accounts he/she creates a StakeTx in order to join the validator set. For account A the attacker generates a random $Seed_{Local}$ as usual. For address B he/she submits the hash of a manipulated $Seed_{Local}$. By brute forcing the fraud is able to come up with a $Seed_{Local}$ that fulfills Condition 4.1 with $TimeInSeconds$ equal to 0.

This shows the possibility with a PoW approach to immediately be reelected after account A has successfully added a block to the blockchain.

Block Height

Therefore, the Condition 4.1 is extended with the $BlockHeight$ parameter as shown in Condition 4.2.

$$\frac{SHA\text{-}256(Seed_{PrevBlock} \cdot Seed_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target \tag{4.2}$$

A node cannot foresee at what block height it is elected to append the next block. Therefore, if the height of the block is included in the PoS condition it becomes computationally more challenging for a malicious user to perform a stake grinding attack.

Such an attack would look as follows. A fraud assumes that he/she is elected to append a block within the next 100 blocks. Therefore, he/she creates 100 addresses that all possess the minimum amount that is required to join the validator set. For each address he/she submits a manipulated $Seed_{Local}$ when joining the validator set. Each $Seed_{Local}$ of the 100 addresses is chosen such that it reelects the vicious user under consideration of an estimated $BlockHeight$.

Therefore, with enough computational power it is still possible to perform a stake grinding attack.

List of Previous Seeds

Hence, the $Seed_{PrevBlock}$ is extended to a list of the previous blocks ($[Seed_{PrevBlocks}]$). If multiple previous seeds are part of the PoS condition, immutable information from other validators are included in the PoS condition.

$$\frac{SHA\text{-}256([Seed_{PrevBlocks}] \cdot Seed_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target \quad (4.3)$$

This makes a stake grinding attack impossible unless the same user has submitted all the considered previous seeds. Therefore, the number of previous seeds must be chosen large enough to prevent such a scenario. The order of previous seeds also matters which makes such an attack highly unlikely. Appending blocks in a controlled order requires a large amount of coins. A fraud would need to be capable of skipping an opportunity to create a block in order to establish the right block sequence.

Therefore, the number of coins needed for such an attack can be controlled by the number of previous seeds in the PoS condition. This number is a system parameter and can be changed by a root account with a ConfigTx. It should be chosen such that the attacker would need to possess at least 51% of the staking coins. With more than 51% of the coins, an attacker could also perform a 51% attack which is simpler to do, since it does not require the computational resources.

Without a Minimum Waiting Time

The minimum waiting time is the number of blocks that need to be passed before a node is part of the validator set after broadcasting the StakeTx (Setting 1). It is also the time that a node cannot validate a block after it has previously added a block (Setting 2).

Setting 1 The vicious node controls two addresses A and B. A is part of the validating set and B possesses the required minimum amount of coins to join the validator set but is not part of it yet. As soon as A is elected to add a block, B broadcasts a StakeTx with a manipulated seed such that the PoS condition for the next block is immediately fulfilled ($TimeInSeconds$ equals 0). This is only possible since the attacker knows all the parameters for the PoS condition for the next block. A includes this transaction in its block and B is immediately elected for the next block. The same setting could be repeated and a single party could take control over the entire blockchain.

Setting 2 Without a minimum waiting time, a malicious node can easily reelect itself. This attack would look like the following. When the attacker is elected for adding a block it finds a manipulated seed by brute forcing such that it fulfills the PoS condition immediately for the next block ($TimeInSeconds$ equals 0). This is possible since the fraud knows all the parameters for the PoS condition for the next block. The manipulated seed is then included the block. As a result every node in the network updates the hashed seed in the attacker's account. Since the seed is manipulated the attacker meets the PoS condition immediately and he/she can repeat the same procedure.

In the case where the minimum waiting time is set to 10 blocks, a malicious user would have to be successfully elected ten times in a row before he can execute such an attack.

In a established system where the staking amount is distributed among many validators, such an attack becomes infeasible. Therefore, the system parameter for the minimum waiting time must be selected accordingly that such an attack becomes unfeasible in the current stake distribution. One also has to consider that a validator is blocked for the number of blocks that defines the minimum waiting time after appending successfully a block.

4.2 Environment

The implemented consensus-algorithm throttles the hashing power of each validator to exactly 1H/s. Compared to specialized Bitcoin mining hardware such as the Antminer S9 which delivers upto 14TH/s [13], the expended resources are negligible and can be carried out by low performance desktop computers.

4.3 Risk of Centralization

This section evaluates the risks of centralization and compares it to PoW implementations. It also discusses the threat in consensus mechanisms where rich nodes get richer.

4.3.1 Mining/Validating Pools

Since the amount of resources used in the implemented consensus-algorithm is insignificant, there is no need for a constant revenue stream for a validator. Therefore, the formation of validating pools as it occurs with mining pools is less applicable.

4.3.2 Cloud Mining/Validating

A validator does not gain a competitive advantage over other validators by buying specialized hardware. Thus, there is no need for centralized cloud service providers to offer hardware rentals which reduces the risk of centralization.

4.3.3 The Rich Get Richer

There are concerns about a threat of centralization in form of ”the rich get richer”. One can argue that a node that owns a large fraction of the tokens will generate more revenue than others and will eventually possess more than 51%.

However, this argument can also be raised in a PoW system. Every time a miner is able to produce a new block, he/she can invest the earned block reward in new mining

hardware. The node that can initially afford the greatest mining infrastructure would also be able to buy new hardware more frequently and eventually become the richest node in the network. Such a node profits from economies of scale and can benefit from higher margins on each block compared to smaller miners. This offers the possibility of buying new hardware even more regularly.

As explained previously in this thesis the block reward and transaction costs in PoS can be lowered drastically compared to PoW due to the lack of burned resources. Thus, it would take a node much longer to possess 51% of all the tokens.

As a conclusion one can argue that such a form of centralization is possible in both types of consensus mechanism. It requires more effort in PoW since one would constantly have to buy new mining hardware and infrastructure. In PoS it takes significantly longer since only a very small portion of the total amount of coins is generated with each new block.

As soon as people start realizing that there is a potential risk of centralization in a distributed system the interest in such a system decreases. As a result the value of the tokens declines with it. In PoS your entire stake is at risk since your investment is in the token itself, whereas in PoW your investment is in form of mining hardware.

If an attacker successfully performs a 51% attack, a minier can move its mining gear on to another blockchain. In PoS it is more difficult to withdraw the investment since the attacker needs to sell all his/her tokens. The market price for the associated cryptocurrency would drop immediately due to the excessive supply and the attacker is probably not able to sell above the price that he/she acquired the tokens. As a result, in PoS systems there is a higher risk of losing a large portion of the investment when attempting a 51% attack and might be more expensive than in PoW.

Chapter 5

Future Work

The consensus mechanism employed in Bazo is unique and does not exist in any other blockchain. Thus, the security of the newly designed protocol must be thoroughly tested.

Although the random validator election process works similar as in PPCoin's protocol, Bazo claims to be less vulnerable towards stake grinding attacks. In a future project the Bazo blockchain should be tested if this claim is justified by attempting to perform such an attack.

Furthermore, a mathematically underpinned analysis is needed in order to define the system parameters securely. For now, the minimum-waiting-time parameter (see Section 3.1.2) can only be adjusted with a ConfigTx signed by a root account. However, this parameter is of great importance in regard to security (as shown in Section 4.1.2). This poses a risk of a security flaw if the responsible root accounts do not adjust this parameter when topology of the validator set changes. A dynamic analysis of the stake distribution among all validators is desired, which also updates the parameter for the minimum-waiting-time automatically within the system.

The same applies for the slashing-window-size and the accepted-time-difference. For the accepted-time-difference the disparity of the clock speeds on different machines running the Bazo protocol have to be investigated.

Chapter 6

Summary and Conclusions

The purpose of this thesis was the technical analysis of existing PoS implementations. Based on the findings the goal was to design and implement a PoS protocol and convert the Bazo blockchain from PoW to the new system. Using a PoS consensus mechanism, the system becomes environmentally friendly and more secure against the 51% attack for a less established cryptocurrency.

The chain-based approach as well as the Byzantine agreement protocols were thoroughly studied. A chain-based PoS protocol approach was chosen out of simplicity of this type of protocol. Several different designs were created and in-depth discussed with the supervisors until a secure design was established.

The implemented random election process is unique and proposes a higher degree of security than the PPCoin implementation. The technical requirements towards the newly designed PoS consensus algorithm were delivered. The evaluation chapter discussed security considerations for the PoS based system.

Bibliography

[1] Bitcoin Energy Consumption Index, https://digiconomist.net/bitcoin-energy-consumption, 10th of November 2017, accessed 10th of November 2017.

[2] Jordan Tuwiner: Bitcoin Mining Pools, https://www.buybitcoinworldwide.com/mining/pools/, 13th of July 2017, accessed 30th of December 2017.

[3] Hashrate Distribution, https://blockchain.info/pools?timespan=4days, 30th of December 2017, accessed 30th of December 2017.

[4] Daniel Cawrey: Are 51% Attacks a Real Threat to Bitcoin?, https://www.coindesk.com/51-attacks-real-threat-bitcoin/, 20th of June 1014, accessed 30th of December 2017.

[5] Jordan Tuwiner: Bitcoin Cloud Mining, https://www.buybitcoinworldwide.com/mining/cloud-mining/, 13th of June 2017, accessed 30th of December 2017.

[6] Jae Kwon: Tendermint: Consensus without Mining, https://tendermint.com/static/docs/tendermint.pdf, 2014, accessed 10th of November 2017.

[7] Vitalik Buterin: Proof of Stake FAQ, https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, 1st of November 2017, accessed 10th of November 2017.

[8] Tendermint Documentation, http://tendermint.readthedocs.io/projects/tools/en/master/introduction.html, 2017, accessed 10th of November 2017.

[9] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, Nickolai Zeldovich: Algorand: Scaling Byzantine Agreements for Cryptocurrencies, https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf accessed 10th of November 2017.

[10] Iddo Bentov, Ariel Gabizon, Alex Mizrahi: Cryptocurrencies without Proof of Work, https://bravenewcoin.com/assets/Whitepapers/Cryptocurrencies-without-Proof-of-Work.pdf, accessed 10th of November 2017.

[11] Sunny King, Scott Nadal: PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, https://peercoin.net/assets/paper/peercoin-paper.pdf, 19th of August 2012, accessed 10th of November 2017.

[12] Livio Sgier: Bazo - A Cryptocurrency from Scratch, 23rd of August 2017, accessed 10th of November 2017.

[13] Jordan Tuwiner: Bitmain Antminer S9 Review, `https://www.buybitcoinworldwide.com/mining/hardware/antminer-s9/`, 27th of June 2017, accessed 30th of December 2017.

Abbreviations

PoW	Proof of Work
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
DoS	Denial of Sevice
AccTx	Creatin Account Transaction
FundsTx	Transferring Funds Transaction
ConfigTx	Changing Configuration/System Parameter Transaction
StakeTx	Staking Transaction
H/s	Hashes per Second
TH/s	Terrahashes per Second - 1'000'000'000'000 hashes per second.
TWh	Terawatt Hour

Glossary

Seed Randomly generated string needed for random election process. User commits to this piece of information by submitting the hash of the seed.

Brute Force A problem-solving system that involves testing every single possible solution to a problem in order to find the right one.

List of Figures

Appendix A

Installation Guidelines

Most of the installation guidlines are taken from the previous system installation instructions [12]. There are new parameters added to the transaction that updates the system config parameters A.2.4. Further, the usage of the new transaction type for joining/leaving the validator set is explained in Section A.2.5 as well as the system keeps track of previously used seeds in case of a rollback A.2.6.

A.1 Validator Application

In order to start the miner application, Go (version 1.8.3 or higher) needs to be installed. The newest version can always be fetched from the following website:

```
https://golang.org/dl/
```

After downloading Go, the environment variables GOROOT and GOPATH need to be set to the corresponding paths. To get the miner application and all necessary libraries, the following command needs to be executed on the command line:

```
$ go get github.com/lisgie/bazo_miner
```

Upon successfully downloading the miner application and all additional libraries, the miner application can be started with the following command:

```
$ bazo_miner <database_file.db> <listening_port>
```

The application consists of two arguments: *database_file.db* points to the storage location of the disk-based key/value store. *listening_port* indicates the port the application listens on for incoming miner and client connections.

A.2 Client Application

To download the client application, execute the following command on the command line:

```
$ go get github.com/lisgie/bazo_client
```

The Bazo client lets a user issue transactions by supplying the necessary arguments for each transaction type on the command line. For a transaction to be successfully validated, it needs to be signed with a private key (this is true for both root and regular user accounts). The keys are stored in regular files and the filename is supplied as an argument.

A.2.1 Key Handling

All signatures in Bazo are based on the elliptic curve digital signature algorithm (ECDSA), a digital signature algorithm based on elliptic curve cryptography. An ECDSA key pair consists of a public key and a private key. The private key is used to sign a transaction and the public key is used for signature verification. Listing A.1 shows an example of a key file:

Listing A.1: Bazo Key File

```
4dc418348a5c77263a70544b49ed07d78714e4df0efe277f4df20cc0a0583717
cec17514ae732cd02e0aff7f6d5125d202fc650b912c489da6ddea0fa153b904
806372a5e4766bd75e9c234280f4879832629f3030b1ebc2cfe12d3b50d6aeac
```

The first two lines make up the public key, the last line is the private key. For transferring funds, only the public key is necessary, the third line can thus be omitted when creating a transaction of this type.

A.2.2 Account Transaction

An account transaction is launched with the following command:

```
$ bazo_client accTx <header> <fee> <privKey> <keyOutput>
```

The meaning of the arguments is as follows:

header Value = 1 creates a new root account, value = 2 removes a root account.

fee The fee to be paid for the transaction. Must be larger or equal than the *Minimum Fee* system parameter.

privKey Points to the key file of a root account (the transaction is invalid if it is signed by a non-root account).

keyOutput File location that stores the keypair belonging to the newly created Bazo account.

A.2.3 Transferring Funds Transaction

A transferring funds transaction is launched with the following command:

```
$ bazo_client fundsTx <header> <amount> <fee> <txCnt>
<fromHash> <toHash> <privKey>
```

The meaning of the arguments is as follows:

header Reserved for future use.

amount Amount to be sent from the sender to the receiver account.

fee The fee to be paid for the transaction. Must be larger or equal than the *Minimum Fee* system parameter.

txCnt This integer parameter is linked to the sender account and needs to be increased with every newly created funds transfer transaction (starting at 0).

fromHash The key file of the sender account (only public key needed).

toHash The key file of the receiver account (only public key needed).

privKey Key file of the sender account (will most likely point to the same location as <*fromHash*>, private key needed).

A.2.4 Changing System Parameters Transaction

A system parameter change transaction is launched with the following command:

```
$ bazo_client configTx <header> <id> <payload> <fee> <txCnt>
<privKey>
```

The meaning of the arguments is as follows:

header Reserved for future use.

id The ID of the parameter to change:

- ID = 1: Block Size [bytes]
- ID = 2: Difficulty Interval [#blocks]
- ID = 3: Minimum Fee [Bazo coins]

- ID = 4: Block Interval [seconds]
- ID = 5: Block Reward [Bazo coins]
- ID = 6: Staking Minimum [Bazo coins]
- ID = 7: Minimum Waiting Time [#blocks]
- ID = 8: Accepted Time Difference [seconds]
- ID = 9: Slashing Window Size [#blocks]
- ID = 10 Slashing Rewards [Bazo coins]

payload The new value to be set.

fee The fee to be paid for the transaction. Must be larger or equal than the *Minimum Fee* system parameter.

txCnt The transaction counter. Won't be checked, this is just to force a new hash for the same transaction.

privKey Points to the key file of a root account (regular users are not allowed).

A.2.5 Stake Transaction

A transaction of this type can be initiated with the following command:

```
$ bazo_client stakeTx <fee> <isStaking> <account> <privKey>
```

fee The fee to be paid for the transaction. Must be larger or equal than the minimum fee set as the system parameter.

isStaking Has to be set to 1 if a node wants to join the validator set or set to 0 if a validator leaves the validator set.

account The key file of the account that wants to join/leave the validator set (only public key needed).

privKey Key file of the sender account (will most likely point to the same location as <*account*>, private key needed).

A.2.6 Seed Storage File

In case of a rollback a validator must be able to restore previous seeds. Therefore, a file called *seeds.json* stores all the previously used seeds. After a rollback a validator reads the hashed seed that is stored in his/her account in the systems state. *seeds.json* maps all the hashed seeds to the actual seeds.

A.2.7 Settings File

Due to the implementation decisions from the previous developer that the name of the key files can be chosen freely, a *settings.json* file was introduce. Its only purpose is to keep track of the file names.

Appendix B

Contents of the CD

- Miner Application Source Code
- Client Command Line Application Source Code
- The LaTeX Source Code
- Final Thesis (.pdf)
- Intermediate Presentation
- Related Papers