University of Zurich[UZH]

Sina Rafati Niya, Burkhard Stiller

BAZO: A Proof-of-Stake (PoS) based Blockchain

TECHNICAL REPORT – No. IFI-2019.03

March 2019

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

ifi

S.Rafati, Burkhard Stiller: BAZO: A Proof-of-Stake (PoS) based Blockchain
Technical Report No. IFI-2019.03, March 2019
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

BAZO: A Proof-of-Stake (PoS) based Blockchain

Sina Rafati Niya, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI,
University of Zürich UZH, Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
Email: [rafati | stiller@ifi.uzh.ch]

***Abstract*—Blockchains (BC) are back-linked chain of records termed as blocks. To establish decentralized trusted systems, BC employ consensus mechanisms. During the past ten years, there have been various proposals of BC design and implementations. However, most of the developed sate of the art BC suffer from high power consumption of miners and low transaction rates in the whole blockchain network. This paper proposes a Proof-of-Stake (PoS)-based blockchain (especially here the BAZO approach), which is designed to enhance efficiency problems of Proof-of-Work (PoW)-based BCs. BAZO enhances the randomness degree in next validator selection in PoS consensus mechanisms at each block height.Having developed the transaction aggregation and double linked blocks, BAZO has enabled the featured for further scalability. Evaluations of this BAZO blockchain outline the efficiency of the design in tackling the 51% attack, double spending, and grinding attacks with avoiding centralization.**

I. Introduction

Blockchain (BC) BCs distribute data storage in terms of records within back-linked lists. Blockchains as decentralized networks of blocks are obliged to offer a sophisticated structure to support different applications. To provide trust in these decentralized systems, various methods of consensus mechanisms have been developed for BC systems. The most used and prominent one is the Proof-of-Work (PoW) of the Bitcoin BC and cryptocurrency [2]. The second most used method is the Proof-of-Stake (PoS) mainly used in Tendermint and many recent BC developments [3]. The most important characteristics of consensus algorithms include scalability, transaction rate, transmission delay, power consumption, security, and privacy, which determine very practical dimensions of BC and their applicability for real applications [9].

consensus mechanisms play a critical role. Consensus between validators on the validity of each transaction in the BC creates trust. Different consensus algorithms are proposed and used to validate transactions toward the accuracy and trust of these systems, such that the coherent blocks of transactions can be reached [3].

Users in BC trust to this system even though they do not or cannot trust other parties of transactions. Users by trusting to those miners or validators can trust to the whole system Users in BC trust to this system even though they do not or cannot trust other parties of transactions.

Proof-of-Work (PoW) is the proof of the attempt each miner puts to mine a new block. The attempt to find a new nonce, with which the hash function produces the output in the range of a requested target is the outcome of many iterations of the hashing function being applied and to solve a cryptographic hash puzzle [12].An important effort is needed to find such a new nonce in the competitive environment of miners. Each miner's success is dependent on its computing power in proportion to the computing power of all miners, thus, leading to a large energy demand. Typically, two financial incentives for miners exist to participate in PoW-based BCs: (a) the fund they receive as a reward of mining a new block and (b) the fund they receive for validating a transaction [14]. A single miner with a limited computational power may not be able to mine a single block in years, that is why usually miners gather and create mining pools. These miners share their information in mining pools to mine a block and eventually, they share the revenues [6].

Proof-of-Stake (PoS) determines a category of consensus algorithms, which provide significant improvements in terms of electricity consumption and scalability over PoW. A PoS consensus algorithm can be described as follows: A validator is a node in the network that validates transactions and adds them to the BC - as done with PoW, too. Any node

in the system can become a validator by depositing tokens of an associated cryptocurrency, which is different compared to PoW. These tokens deposited cannot be spent as long as the validator is part of the validator set. Thus, these deposits are used to punish malicious nodes, since they have not check the validity of previous transactions in previous blocks. Validators are elected proportionally to the funds deposited for appending the next block to the BC. The right to add the next block is determined in a decentralized selection procedure depending on the number of tokens deposited by each validator. This category of PoS consensus mechanisms consists of many different algorithms and protocols, where each offer the tasks of a random validator's election and a reward system, all these challenges addressed differently.

Some of the problems experienced with various aspects of using PoW-based BC regarding cost and time deficiencies are compared with PoS as follows:

Environmental Harm: Digiconomist [1] reports that the Bitcoin BC consumes more than 36 TWh annually. This amount is more than the entire nation of Bulgaria consumes every year. In other terms, the Bitcoin BC uses as much electricity as more than 3 million households together.

Thus, in order to maintain the Bitcoin BC the global mining costs amount to over 5 million USD every day.In contrast, there is no need for an extensive computation task in a distributed system using a PoS consensus mechanism in order to reach a comparable level of security.

Detailed explanation of all sections in this work is provided in [13] and here, due to the page number limitations, only the most important aspects are covered.

Risk of Centralization in Form of Mining Pools: The chances of adding a new, valid block for an individual miner with a PoW-based BC is extremely low [10]. Therefore, miners can join mining pools, where their computational power is collaboratively deployed. However, if these mining pools joined together, the Bitcoin BC would be vulnerable to a 51%- attack, where these mining pools could always provide a longer competing chain than the currently accepted one [5]. Therefore, the Bitcoin BC would be controlled by a single centralized entity.In contrast, the need for a constant revenue stream is not required in a PoS system, since the validator does not have to be compensated for resources burned. Therefore, the formation of large validator sets is less likely.

Risk of Centralization in Form of Cloud Mining: Cloud-mining allows people to mine cryptocurrencies without owning mining hardware [19]. Companies providing cloud-mining services profit from the economies of scale, such as special deals on hardware orders and lower maintenance costs. Therefore, these companies gain a economic 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.

Entry Barrier: A miner in a PoW-based system, including specialized hardware, gains an advantage over the competition, whereas in a PoS-based 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-based system is, therefore, significantly lower than becoming a miner in a PoW-based system.

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 [10]. Practically, any machine with a sufficient amount of storage and bandwidth can be part of the validator set. Therefore, the community is less split up as it occurs to be in PoW-based systems.

51% Attack: In the initial phase of a new BC the number of validators/miners is limited. This poses a great risk of a 51% attack, indicating that 51% of all miners collude. A malicious user can 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 remainder of the network. Since the BC protocol always follows the longest competing chain, therefore, the malicious user has control of the entire BC. The developers of a PoS-based system can hold back 51% of all coins until the network is established in order to prevent such an attack by an outsider.

Transaction Fees: Transaction fees – determining the incentives to persist a block in a given BC – prevent a BC from being spammed, but also

compensate miners for their computational effort and electricity costs. However, no resources are burned within a PoS consensus mechanism and, therefore, the transaction fees should be lowered significantly.

The presented work here covers the design of the PoS-based BAZO BC. Thus, BAZO increases the security of the system against the risks of a 51%-attack. Moreover, the scalability and BC size are considered and transaction aggregation methods are developed. Since it is very difficult to implement a fully decentralized PoS, the form of a delegated PoS (DPoS) or a PoS/PoW combination scheme is considered for BAZO, still reaching those targets.

This paper is organized as follows, Section **??** explains the PoS and PoW consensus mechanisms with the focus on the problems of PoW which could be addressed by PoS BC.Also, the challenges of developing PoS-based BC are explained in this Section. Section II introduces the PoS based proposals of related implementations. In Section III design of the proposed PoS BC is described and evaluations done in Section V. At the end of this paper, concluded points are presented in the Section XI.

A. PoS Challenges

key challenges to be met while developing PoS consensus mechanisms include (a) mechanisms needed to prevent nodes from double spending, (b) manipulation of the random election, (c) DoS attacks, and (d) the difficulty of keeping validators online at all times.

1) Nothing-at-Stake (Double Spending) Problem: In the case of BC forks, which sees multiple side-chains competing with each other at the same time, a validator needs to decide on two or more options, where to add the next block. Unlike in PoW, PoS does not ”burn” resources in the process of finding a random node in the network. This allows the validator to append a block on any of the competing chains. Therefore, it is a potential validator's 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 finally.

If all validators act economically and append their block on every competing chain, the blockchain will never reach consensus, even if there exist only honest validators. Figure 1 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, PoS allows for potential misuse. Since for PoW the hash of the previous block is included in the calculation, as shown in Figure 2, a miner will always put all his/her mining power into the longest chain, which is the chain that will most likely be accepted by the network.

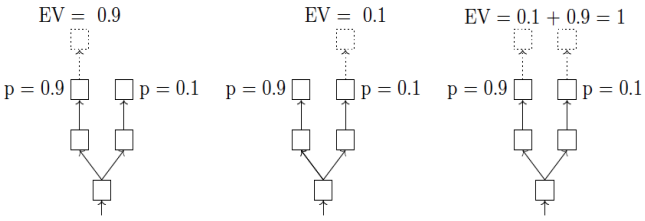

Fig. 1: Competing Chains in PoS [4]

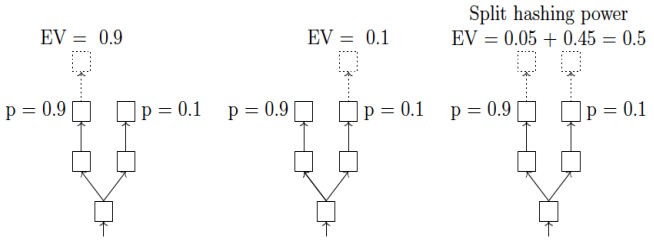

Fig. 2: Competing Chains in PoW [4]

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 as well. 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

miners and, therefore, p has to add up to exactly 100 percent over all competing chains of all 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. his property is described in the following scenario and shown in Figure 3. 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 BC. Therefore, every following block is added on both chains, too. With the assumption that validator F has a staking power of 1% and validator G of 2%, all the validators between block f and y make up for the total stake of 96%. At the time of adding 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, well above 1.

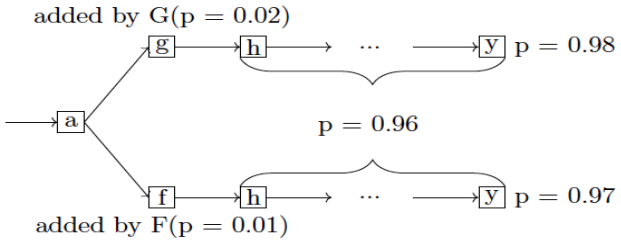

Fig. 3: Staking Weights in PoS [4]

A malicious user Z can take advantage of such a situation and double spend his/her 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 that 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 (cf. Figure 4). In this example the malicious user has double spent his/her coin with only 2% of the total stake. To ensure a secure implementation of PoS, a PoS consensus protocol has to implement a scheme allowing validators to be punished for appending new blocks on multiple chains at the same height.

2) Grinding Attacks: If validators were able to manipulate the random election process of a consensus algorithm, grinding attacks happen. 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 or trying many parameters, which 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 needs to ensure that 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 the Section III-A1.

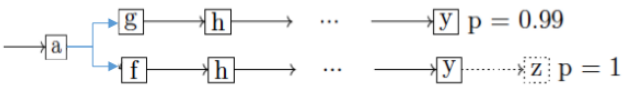

Fig. 4: Double Spending in PoS [4]

3) Denial-of-Service (DoS) Attacks: 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 Denial-of-Service (DoS) attacks. Therefore, it is advantageous to run the election process privately. In a PoW protocol, the election process is done in a way that each validator independently tries to find a solution to a mathematical puzzle. These puzzles are 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 single node in the network. The same level of unpredictability in the election processes as in PoW is desired for a PoS protocol.

4) Availability: Many different parties competing for the next block decrease 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 Peercoin implementation in Section II-1.

A protocol that includes coin aging 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 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 should incentivize validators to be online at any given time.

II. Related Work

The two different types of PoS consensus mechanisms of major importance currently existing are "Chain-Based Protocols" and "Byzantine Agreement Protocols". 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.

1) Chain-Based Protocols: In this category a validator is chosen randomly from the validator set in order to append the next block. This random election process solely depends on information that is stored within the BC, such as the "height" of a particular block or the amount of coins a validator possesses. A selected instance of this approach is Peercoin.

Peercoin: Peercoin or PPCoin assigns the right of appending a next block to a validator that is in compliance with the condition (1) determined below [7]. 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 validator. *Target* is a constant value that determines the speed of the BC. *Coins(TxOutA)* represents the amount of coins that a validator has deposited into the unspendable address.

$$\frac{hash(PrevBlockData \cdot TimeInSec \cdot TxOut)}{Coins(TxOut) * TimeWeight(TxOut)} \leq Target \tag{1}$$

Peercoin uses "coin aging" [18] to prevent stakeholders with a significant fraction of all 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 Peercoin limited the maximal amount of *TimeWeight(TxOut)* to 90 days in their version v0.3 protocol [7].

Peeroin's random election process is comparable to 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 which hardware is being used. In turn, Peercoin is vulnerable to stake grinding attacks, where a validator can influence parameters in order to reach a higher probability to be re-elected for the next block. This is possible, because the data of the previous block is included in the PoS condition [4].

2) Byzantine Agreement Protocols: In this category 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 his/her stake in the system. In each round validators can commit to a block. If they agree to the block proposed and if the block receives a majority of votes, it is considered to be valid. All messages that are used for proposing a new block or committing to a block are handled by a gossip protocol. This means none of these messages are recorded on the blockchain. A selected instance of this approach is Algorand.

Tendermint: In Tendermint (cf. Figure 5) 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*. 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. Every validator has to make a decision during "Prevote" 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* [11]. In the "Precommit" step, each node validates if it received more than $\frac{2}{3}$ of the total *Prevotes*. In this 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.

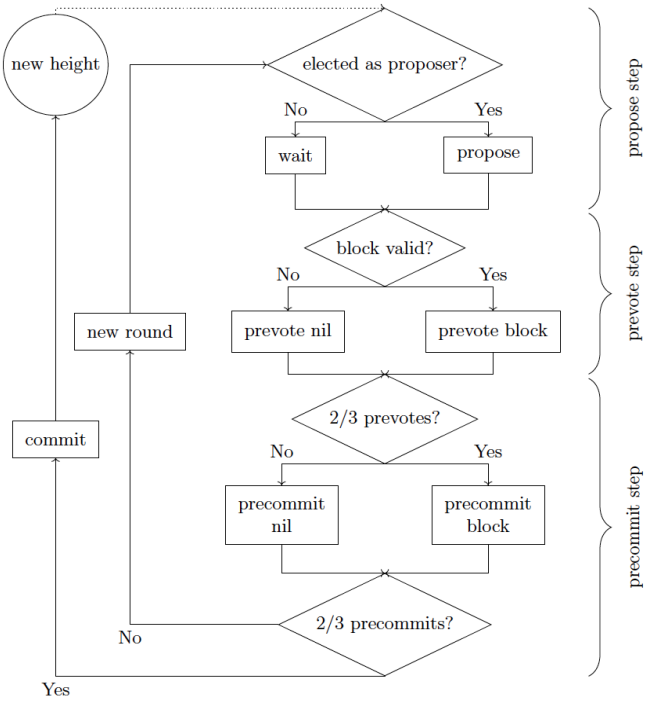

Fig. 5: Tendermint State Machine

Algorand: Algorand's PoS protocol is similar to Tendermint with significant improvements. While in Tendermint the right of proposing a next block is being passed in a round-robin fashion [11], Algorand uses a mechanism based on verifiable random functions that allow nodes to privately check whether they are part of the current round. After gossiping a message to the network, these nodes are immediately replaced. The mechanism developed in Algorand, prevents Denial-of-Service (DoS) attacks on validators chosen after their identity is revealed. Algorand addresses the scalability problem by randomly selecting a committee, which determines 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 [20].

III. BAZO Design

The previously existing PoW-based BAZO BC [17] was revised and extended to a PoS-based BC, which is referred as BAZO in this paper below [16]. The proposed BC is explained here with elaborations on Transactions, Accounts, Block, PoS Condition, and validations.

In BAZO, the chain-based protocol is developed for simplicity reasons in which the PoS protocol works comparably to the PoW consensus mechanism with key differences. It designs a throttled PoW algorithm. Since a validator is limited to exactly 1 h/s (hash per second). BAZO defines a semi-synchronous system, such that every node has its own local time. If a node attempts to speed up his hashing power, the system detects the malicious node and the suggested blocks are rejected. BAZO chooses validators proportionally to the number of coins that each validator owns.

Furthermore, seeds are replaced by RSA scheme [15] to enhance the randomization of selecting the next validator at every Block Height (BH) (cf. Section III-D). One key advantage of this PoS protocol is that a validator is always elected unless all validators are offline at the same time. Other PoS protocols would need to implement a fall-back mechanism that deals with a deadlock, which occurs, if all members of a chosen subset of validators are offline.

A. Transactions

Transactions are used to change the state of accounts in BAZO. Three transaction types are already existing in the original PoW-based BAZO, which include the "account creation", "transferring funds", and "adjusting system parameters" [17].

1) Stake Transaction (StakeTx): A node in the active validator set confirms transactions proportionally to their stake in the form of blocks of transactions. This new type of transaction allows a node to join or leave the validator set. This transaction consists of the following parameters:

Fee: The fee is a payment as 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 the PoW-based protocols, the validator that appends the next block will be rewarded with transaction fees of all transactions within the block appended. Fee amounts are smaller compared to a PoW-based system, since a validator does not have to be compensated for electricity costs. However, a small amount – which needes to be calculated via configuration transactions – still has to be paid for every transaction in order to prevent to BC from being spammed.

Is Staking: This defines a Boolean state, whether the node wants to join or leave the set of validators.

Account: The account is defined by the hash of the public key of the issuer.

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 (cf. Section III-A2). Staking transactions are referred to as *StakeTx*.

key Commitment: Each node has to generate a Public key Pk, Private Key Sk, (Pk,Sk) pair. For this purpose the RSA algorithms is chosen with a key size of 4096 bits [15].

2) System Parameters (ConfigTx): With *ConfigTx* system parameters can be adjusted without the need of a hard fork. The configuration transaction of the PoW-based BAZO BC is extended by the following parameters to reach a PoS-based BAZO:

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 BC, a validator must also wait the same number of blocks before another one can be added. The reason for waiting a certain number of blocks is the prevention of a stake grinding attack (cf. Section V-A2).

Accepted Time Difference: An important characteristic of a semi-synchronous system is the different clock speed in every node participating within 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 this BC.

Slashing Window Size: As described in Section I-A1 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: 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) Funds Transaction (FundsTx): transfers funds from one account to another. Any transaction from an active validator will be rejected, if the balance of the node is below the minimum staking amount after the transaction.

B. Account

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

Is Staking: This Boolean parameter describes, whether the account is currently part of the active validator set or not.

Staking Block Height (SBH): The system registers the block height at which an account joined the set of validators. This parameter is needed for the slashing condition.

C. Block Structure in Bazo

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 (cf. Section III-D).

Height: The height of a block refers to the number of previously appended blocks to the BC. This parameter is needed for the PoS condition (cf. Section III-D) as well as the slashing condition (cf. Section III-E).

Commitment Proof: This parameter represents the output of $RSA(SK, SHA3-512(H))$. SK represents the private key that corresponds to the public key PK that was set in the initial *StakeTx* pf the node. PK can be used by other validators to verify the proof. H is the Height at which block created.

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.

D. 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 has passed, it is eligible to append blocks to the BC. For that aim, it must provide a valid *TimeInSeconds (T)* for the PoS condition 3. Each of the parameters provide an important property:

$$\frac{SHA\text{-}256([Seed_{PrevBlocks}] \cdot Seed_{Local} \cdot BH \cdot T)}{Coins} \leq Target \tag{2}$$

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 BC by broadcasting the proposed block. The seed is used for the random election process for the upcoming block. As highlighted in Section II-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 **??** stake grinding attacks are revisited and evaluated.

$$\frac{SHA\text{-}256([P_{PrevBlocks}] \cdot P_{Local} \cdot BH \cdot T)}{Coins} \leq Target \tag{3}$$

List of the Previous Proofs ($Proof_{PrevBlocks}$): By having the list of previous proofs at hand, a stake grinding attack becomes unfeasible [15].

Local Proof (P_{Local}): With Local Proof, even a validator with a low amount of coins can append a block to the BC [15].

Block Height (BH): The height of a block is characterized by the number of previously added blocks in the BC 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 the difficulty in the PoS protocol can be adjusted with this global variable in order to determine the speed of the BC. 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 [17].

E. Validation

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

Commitment Proof: Each validator checks the origin of message by checking the public key of the

validator in the *Commitment Proof* of the new block [15].

Minimum Waiting Time: The minimum waiting time requirement is needed in order to prevent stake grinding attacks (cf. Section V-A2). 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. **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.

F. Competing Chains

Due to latency and the nature of the PoS condition it is possible that BC forks happen and validators compete on two or more chains. In such a case the proposed BAZO protocol follows the same principle as in a PoW system which covers the following possibilities:

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 Competing Chains.

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 [17].

IV. Scalability Enhancements with Transaction Aggregation

The idea behind transaction aggregation is to aggregate valid transactions such that multiple transactions from one sender or to one receiver are visible as one transaction in the BC. This should enhance the *TPS* because more transactions can be validated in one block. Thus, the block size becomes less of a limiting factor when many transactions with the same sender or receiver are issued to the network. Furthermore, transaction aggregation reduces the BC's overall size because fewer transactions are visible in the BC. Especially when aggregating transactions, which are validated in an already closed block, the overall BC size can shrink since at a certain point these blocks can be emptied completely. This results in a reduced block size.For the aggregation, a new type of transactions, called *AggTx*, is introduced in Bazo. Furthermore, all funds transactions are updated accordingly.

An Example: User *A* sends 2 BAZO-Coins to *B* and 5 Bazo-Coins to *C*. Without transaction aggregation both transactions are listed in a block as (schematic) *(A →B : 2) & (A →C : 5)*. With transaction aggregation only one transaction *(A →[B,C] : 7)* will be written into the block. This illustrates that the overall BC size could be smaller and more transaction can be handled because they are aggregated. The aggregation is planned to be fully hidden from the users.

Aggregated: The variable 'Aggregated' is a *boolean* and does indicate if this transaction is aggregated already.

Block: In this filed the hash of the block, in which this transaction is validated the first time, gets stored. This is used for rollback scenarios. It is of type *[32]byte*.

1) AggTx: This type of transaction does aggregate and sum up matching funds transactions and older aggregation transaction. Instead of these transactions, an *AggTx* will be listed inside a block. **Amount:** The amount is the summed up amount of all transactions aggregated inside this *AggTx*. It is of type *uint64*. **Fee:** The fee of this transaction. It is set to 0 because, at the time of writing, the users should not be charged for this type of transaction. It is also a *uint64*. **From:** It is a slice where the addresses of

all senders sending a transaction aggregated in this block are stored. It is a slice of type *[][32]byte* **To:** This is the counterpart of the *From* field and filled with the addresses of the transactions' receivers. It is also a slice of type *[][32]byte*.

AggregatedTxSlice: This slice is of type *[][32]byte* and does store all hashes of the transactions aggregated inside this *AggTx*.

Aggregated: The variable 'Aggregated' is a *boolean* and does indicate if this transaction is aggregated. **Block:** In this filed the hash of the block, in which this transaction is aggregated the first time, gets stored. It is of type *[32]byte*. **MerkleRoot:** Root of the Merkle tree to ensure integrity and the correct order for the transactions aggregated in this *AggTx*. It is a slice of type *[32]byte*.

These *AggTx* are added to the blocks similar to other transactions. They are listed in the $AggTxData$ slice.

2) Theoretical Aggregation Process: Funds transactions and aggregation transactions can be aggregated in two different ways. All other types of transactions are not taken into account. Furthermore, it is not possible to combine an already aggregated transaction aggregated by the sender and one by the receiver. This results in either the *From* or *To* slice to have a length of one.

Transactions can be aggregated by the sender. Then all transactions which are sent by a specific wallet are aggregated into one *AggTx*. When aggregating transactions this way, the *From* slice has a length of one, as there is only one sender included. On the other hand transactions can also be aggregated by the receiver. Then all transactions sent to one specific wallet are aggregated into one *AggTx*. Here the *To* slice is only length one because all transactions are sent to one specific receiver.

When a miner combs through all open transactions he tries to aggregate as many open funds transactions as possible according to these two rules. If two or more transactions can be aggregated, their transaction hashes are written to the *AggTx's AggregatedTxSlice* and the transactions' boolean *Aggregated* will be set to *true*.

In the next step, the miner checks already closed blocks, whether there are transactions (either *FundsTx* or *AggTx*) which do match the chosen pattern (either aggregated by sender or receiver) and are not aggregated by now. If such historic transactions exist, they are also added to the *AggTx*. But they do not have an influence on the state during the post validation of a block anymore.

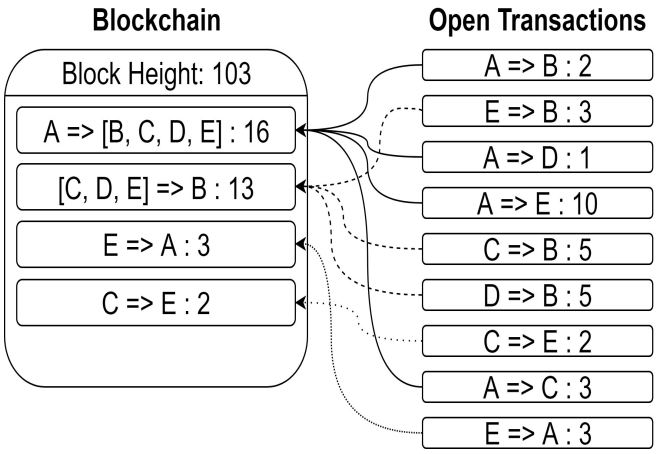

Fig. 6: Transaction Aggregation Concept

In figure 6 the aggregation process is visualized schematically. The letters are wallets and the numbers are the amount of BAZO-coins sent. All open transactions are listed on the right side. In this example, the historic aggregation is omitted due to simplicity and only one of various possibilities is shown.

An algorithm group these transactions in an optimal way, such that the fewest transactions are listed in the block, but the most transactions are validated. Without Transaction Aggregation, all these open transactions try to be in current block 103. When the block size is assumed to be limited to five transactions, with aggregation, there is still place for one more transaction whereas without transaction aggregation not even all nine transactions can be validated in the current block. Because BAZO only writes the hashes of transactions inside its blocks, this is possible. Consequently, with aggregation only the hashes of the two aggregated transactions and the two normal funds transactions, which cannot be aggregated in this block, are stored in the block's body.

Furthermore, it is also visible that it actually does not matter, how many transactions are aggregated in one *AggTx*. If *A* would have sent more transactions, still only one transaction is written into the block, but this transaction would aggregate more transac-

tions. This is especially nice for a BC which is used in a case, where often multiple transactions are sent from one or to one peer. With small modifications, an imaginable example use case would be a BC which stores values sent from Internet-Of-Things devices always to the same receiver.

The miners do not earn a specific fee for aggregating. But they still receive all the fees which belong to the *FundsTx*. This results in miners that want to validate as many *FundsTx* as possible and thus earning as much as feasible. The more transactions they can aggregate the more transactions are in a block and they will get a higher reward. Since the block's size does not grow with every transaction, they can add more transaction into one block.

A. Double Linked Blockchain

The concept of a double linked BC is a result of the idea to remove all transactions from a block once all of them are aggregated in a later validated block. This is kind of a contradiction against the theory of a BC where all validated blocks are immutable. They are unchangeable because it would take too much effort to recalculate the complete chain since this adapted block, and additionally persuade over 50% of all miners to accept the newly created blocks.

In BAZO the block hash is calculated from various block related input fields. One of these variables is the *Merkle root*, which ensures transaction verification. It ascertains that transactions neither can be added to or removed from a block nor the ordering can be changed once a block hash is created [**?**]. Thus, the removing of transactions is only possible when a new additional block hash is calculated for every block because the old hash is becoming invalid, as soon as some transactions are removed. This new hash, called *HashWithoutTransactions*, is used always when the normal hash is becoming invalid. The normal hash becomes invalid because the *Merkle root* changes.

The goal and also the specification of the double linking is, that at least one link to the previous block is valid. Additionally to the common variables included in a block, the following fields are added in respect of the double linking of the BC:

Aggregated: It indicates if a block is aggregated and therefore does not contain any transactions anymore. It is of type *boolean*.

HashWithoutTransactions: This hash is used once all transactions from a specific block are aggregated. It can be calculated when not taking the transactions into account and as a consequence assuming an empty block. Thus, it is only possible to empty a block, once all transactions are aggregated and removed. It is of type *[32]byte*.

PrevHashWithoutTransactions: This field links the current block to the previous one once all transactions in the previous block are aggregated.

ConflictingBlockHashWithoutTx1: HashWithoutTransactions of the first conflicting block.

ConflictingBlockHashWithoutTx2: HashWithoutTransactions of the second conflicting block.

1) Theoretical Double Linking Process: In figure 7 the concept of a double linked BC is illustrated. Every block, expect the genesis block, can either be in the storage *Blocks With Tx* or *Blocks Without Tx*. This two versions of a block are indicated with *block-name$_{w/}$* (including transactions) and *block-name$_{w/o}$* (without transactions, meaning this block never contained transactions or all of them are aggregated by now). The increasing block number indicates which block is the ancestor, and the arrows point to them.

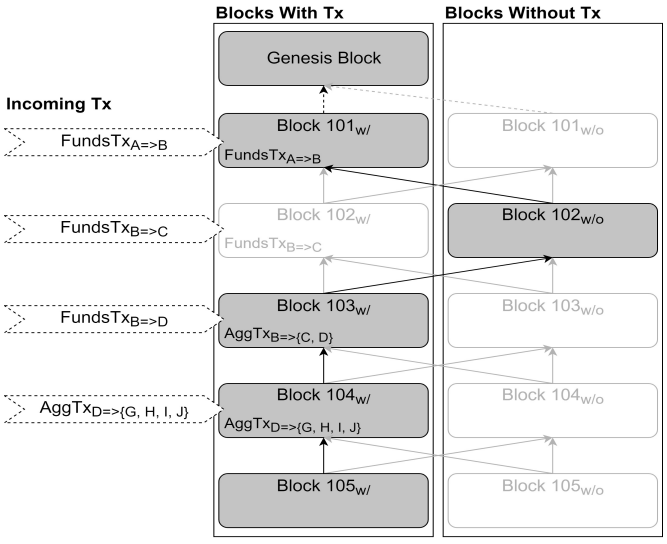

Fig. 7: Double Linked Blockchain Concept in BAZO

Every block, expect the genesis block, can contain transactions. These transactions are sent from clients to the network, what is indicated on the left side, as *incoming Tx*. Transactions are read as

$FundsTx_{senderAddress=¿receiverAddress}$ or when aggregated as $AggTx_{senderAddress=¿\{receiverAddress\}}$. This happens as a historical aggregation in block 103 or while mining a new block denoted as the incoming *AggTx* also included in block 104.

As $FundsTx_{B=¿D}$ reaches the network, the miners search in already validated blocks for other *FundsTx* or *AggTx* with either the same sender or receiver. In this example $FundsTx_{B=¿C}$ in block 102 can be aggregated, which leads to the case where in block 102 all transactions are aggregated.

Once all transactions are aggregated and the block is out of the exclusion zone, it can be transferred to the storage without transactions. The exclusion zone is defined as the *current blockheight* minus *NO_EMPTYING_LENGTH* what ensures that the user-defined *NO_EMPTYING_LENGTH* last blocks are not moved even though all their transactions are aggregated. Meaning only blocks with a blockheight smaller than *currentBlockheight* - *NO_EMPTYING_LENGTH* are moved. Block 105 is not emptied yet despite the fact it does not contain any transactions. Once block 105 will be out of the exclusion zone, it will be transferred to the *Blocks Without Tx*. When a *NO_EMPTYING_LENGTH* of *2* is assumed, block 105 can be moved once a block with height 108 is appended to the chain.

When emptying block 102, it will be moved to the *Blocks Without Tx*, and the hash *HashWithoutTransactions* gets valid. Therefore block 103 is not linked to block 102 via the *Previous Hash* anymore but over the *PrevHashWithoutTransactions* now. It has to be ensured that one link between two consecutive blocks is always valid. In figure 7 this is indicated with the black arrows between blocks. This results in a valid chain indicated with grayish background color whereas all other blocks are only there for visual purposes and therefore slightly faded.

V. Evaluation and Simulation Results

Evaluations of BAZO is performed by setting up a test-bed including 9 miners hosted on Amazon Web Services (AWS) located in Oregon, São Paulo, Ohio, London, Paris, Mumbai, Singapore, Tokyo and Sydney. In this network more than 2200 transactions monitored between the clients and miners. The maximal block size is set to 500'000 Bytes. The block interval, which defines the time span between two blocks added to the chain is set to 180 s and the difficulty interval is set to 20 blocks. This means that the difficulty is checked all 20 blocks and if blocks are mined too fast or too slow the target gets adapted accordingly to match the defined 180 seconds in future. Related information can be seen in the Table I.

In the following part evaluations regarding the three major metrics of (a) security challenges, (b) environmental side effects, and (c) risks of centralization are elaborated.

A. Attack Countermeasures

The attack scenarios and their countermeasures in order to mitigate them elaborates explicitly on Denial-of-Service (DoS) and stake grinding attacks.

1) Countermeasure on DoS and Stake Grinding Attacks: The election process of the BAZO BC is done in the same private manner as done within PoW consensus algorithms. By increasing the randomness in the selection of next validator, attackers wont be able to predict the next validator. This will cause in protection of the system against DoS and Grinding attacks.

2) Without a Minimum Waiting Time: The minimum waiting time here explained in two settings.

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 it 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 (T = 0). This can only be possible, if an attacker knows all 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 BC.

Setting 2: Without a minimum waiting time, a malicious node can easily reelect itself. This attack reads as follows. 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 (T = 0). Therefore, the system parameter for the minimum waiting time must be selected accordingly, that such an attack becomes unfeasible in the current distribution of

TABLE I: BAZO Simulation Results

Miner Location	#Blocks Mined	First Block	Last Block	Timespan	Elapsed Time (s)	Blocks Per Min	#Blocks Added To Chain	Total #TX sent	Total #TX Validated	Average Number of TPB	Max TPB
London	51	11:44:05	18:58:04	07:13:59	26039.88	0.12	43	21536	21536	500.83	4403
Ohio	41	11:38:45	18:56:00	07:17:15	26235.14	0.1	43	9952	9952	231.44	2272
Tokyo	32	11:23:02	19:09:54	07:46:52	28012.53	0.07	43	17589	17589	409.04	5204
Sydney	77	11:14:54	18:53:50	07:38:56	27536.54	0.17	32	13159	13159	411.21	2272
Mumbai	69	11:17:11	19:06:23	07:49:12	28152.95	0.15	43	16908	16908	393.20	2272
Sao Paulo	51	11:33:17	19:16:45	07:43:28	27808.23	0.12	43	16942	16942	394	2174
Singapore	58	11:23:13	19:21:46	07:58:33	28713.40	0.13	43	10104	10104	234.97	3090
Oregon	42	11:18:50	19:18:46	07:59:55	28795.47	0.09	32	13727	13727	428.96	3836
Paris	54	11:29:18	19:28:02	07:58:44	28724.73	0.12	40	15927	15927	398.17	3302
Overall	475	11:14:54	19:28:02	08:13:08	29588.98	0.97	362	135844	171230.03	———	5204
Average per Miner	52.78	11:26:57	19:09:57	07:42:59	27779.87	0.12	40.02	15093.78	19025.55	473.02	578.77

stake. 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.

In 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/she can execute such an attack. In an established system, where the staking amount is distributed among many validators, such an attack becomes infeasible.

B. Environmental Side Effects

The consensus algorithm implemented for BAZO throttles the hashing power of each validator to exactly 1H/s. Compared to Bitcoin mining hardware, such as the Antminer S9, which delivers up to 14TH/s [8], the expended resources are negligible and can be carried out by low performance desktop computers.

C. Risk of Centralization

Mining and Validating Pools: Since the amount of resources used in the BAZO is low (cf. Section III), formation of validating and mining pools is less applicable as there is no need for a constant revenue stream for a validator.

Cloud Mining/Validating: In BAZO a validator does not gain a competitive advantage over other validators by buying specialized hardware (cf. Section III). Thus, there is no need for centralized cloud service providers to offer hardware rentals, which reduce the risk of centralization.

The Rich Gets Richer The rich get richer problem corresponds to a node owning a large fraction of tokens and which will generate more revenue than others, eventually possessing more than 51%. Thus, such a form of centralization is possible in PoW and PoS consensus mechanism. Since the block reward and transaction costs in PoS can be lowered drastically compared to PoW due to the lack of burned resources, it would take a node much longer to possess 51% of all tokens. It requires more effort in PoW, since constantly new mining hardware and infrastructure had to be added. In PoS if the reward of validators paid proportionally, since only a very small portion of the total amount of coins is generated with each new block it takes significantly longer than PoW.

While this may happen, the exact same 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 becomes the richest node in the network. Such a node profits

from the 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 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 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 miner can move its mining gear on to another BC. 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.

In this section, the results from the performance analysis are discussed. Therefore, the test cases defined in chapter **??** are used. There is a section about the block size, about the interval between blocks and about the blockchain's overall size. The chapter closes with the benefits and obstacles of transaction aggregation and a prospect about possible future work.

VI. Different Block Sizes

Here, the evaluation and comparison between different block sizes and its influence on the *TPS*, with an initially defined amount of transactions sent to the network, is measured and listed. However, it is possible that not the same amount of transactions are sent to the network in each test run because of either the miner or the client crashes. This is mainly caused by an unrecoverable *SIGBUS* error. For certain test runs, the number of transactions sent is lowered on purpose, as described in chapter **??**.

Table II does show the transactions per second rates for test cases with transaction aggregation enabled whereas tableIII shows the test cases without transaction aggregation. In both tables, the block size has unit *byte*, and the values belonging to transactions have unit *transaction* or *transactions per second*.

The actual block size (*ABS* and unit *byte*) is the available space in a block where transactions can be filled in. At the time of writing, it is $DefinedBlockSize - 658byte$. The *658 bytes* are used for block related values and therefore this space cannot be filled with transactions.

At a first look, it occurs strange that there are also TPS_{min} and TPS_{max} beside the normal *TPS*. This is due to the fact, that some miners need longer to fetch all transactions when they do not receive all transactions during broadcasts, as described in section **??**. Also nearly always the virtual machines hosted on GCP had a lower *TPS* than the ones on AWS, what may be an indicator that they are slightly underpowered in regards of the random access memory. The TPS_{max} is an evidence of what speeds can be possible with transaction aggregation as it is implemented in this thesis. However, in chapter VII, it is visible that also with transaction aggregation, the TPS_{min} and TPS_{max} can be close to each other. Therefore it is strongly assumed, that this difference is caused by connection issues, which results in requesting more transactions.

As visible in table II, the block size does not take a big influence on the reached *TPS* value. This does have multiple reasons:

Transaction Aggregation: Since transactions are aggregated, not every transaction needs space in a block, and thus, more transactions fit into one block. Actually it does not matter, if there are two transactions from *A* to *B* or if there are one hundred transactions from *A* to *B*. With transaction aggregation, only one transacting will be written into the block. Consequently, it is good to group as many transactions as possible and bad, not to group any transaction, because it needs too much space in comparison.

Unlimited *AggTx* Size: If a transaction does not fit in one block, it is likely that it is validated in the next block because there is no limit in how many transactions can be written into one *AggTx* and because of the *splitAndSort*-algorithm's design (explained in section **??**, algorithm **??**). This algorithm takes the senders or receivers which have most awaiting transactions to be validated. Thus, the more transactions from one sender or to one receiver are

TABLE II: Table of different block sizes influencing the *TPS* with transaction aggregation enabled.

Block size		Transactions						
Defined	ABS	#sent	#validated	TPS_{sent}	TPS	TPS_{min}	TPS_{max}	$\frac{TPS}{TPS_{calc.}}$
1'000	342	179'328	178'196	33.1	28.0	14.6	32.7	39.2
5'000	4'342	181'933	181'933	33.3	28.5	20.9	33.2	3.1
20'000	19'324	181'613	181'613	33.0	28.0	16.7	32.9	0.7

TABLE III: Table of different block sizes influencing the *TPS* with transaction aggregation disabled.

Block size		Transactions						
Defined	ABS	#sent	#validated	TPS_{sent}	TPS	TPS_{min}	TPS_{max}	$\frac{TPS}{TPS_{calc.}}$
1'000	342	19'000	19'000	36.3	0.6	0.6	0.6	0.7
5'000	4'342	74'960	74'834	34.7	7.0	7.0	7.0	0.8
20'000	19'342	181'078	181'078	33.5	27.3	27.2	27.4	0.7

in the mempool, the more likely it is that they get validated. Because the transactions from / to one wallet, which cannot be validated in a block, are still in the mempool, it is likely that there are more transactions matching the selection criteria for the next block. Since the size of an *AggTx* is unlimited, they are able to aggregate as many transactions as possible.

Test Case: In the test cases, there are maximal 20 users in the network. Thus it is likely, that the same sender or receiver is found quickly. This does help to keep the *TPS* at a high level. It is obvious that transaction aggregation works better the more similar sender or receivers are in the transactions. But since BAZO is planned to become an IoT blockchain, where multiple IoT devices send their data to one receiver, this limited number of wallets is not a problem.

Transaction Sending: Transactions are sent approximately every 12 second. If transactions would only be sent after the previous transaction is validated in the network, aggregating by sender would not work. The current acknowledgment a miner sends to a client is only confirming that a certain transaction is sent to the network. A client does not know if and when the sent transaction is validated. Thus, aggregating by sender is possible.

The $\frac{TPS}{TPS_{calc.}}$ does indicate by which factor the aggregation increases the maximum possible $TPS_{calc.}$ value. Thus with a block size of 1'000 byte, the version with transaction aggregation can theoretically handle roughly 39 times more transactions than without aggregation. It is clear, that with a smaller $TPS_{calc.}$ and a constant *TPS*, this factor increases. The $\frac{TPS}{TPS_{calc.}}$ for a block size of 20'000 byte is below one, because theoretically over 600 transactions fit into a 20'000 byte block what results in a $TPS_{calc.}$ of about 40 transactions. This $TPS_{calc.}$ is higher then the TPS_{sent} and therefore, the ratio cannot be bigger than one. Summarizing this, it is not possible to generally say, that with transaction aggregation it is exactly 39.2 respectively, 3.1 times faster, because this numbers are highly influenced by the actual *TPS* which can not be higher than the TPS_{sent}. Thus, sending more transactions probably increases the *TPS*, and therefore also the $\frac{TPS}{TPS_{calc.}}$ ratio, even more. In table III, the fraction $\frac{TPS}{TPS_{calc.}}$ can be understood as degree of utilization in terms of defined and actual block size.

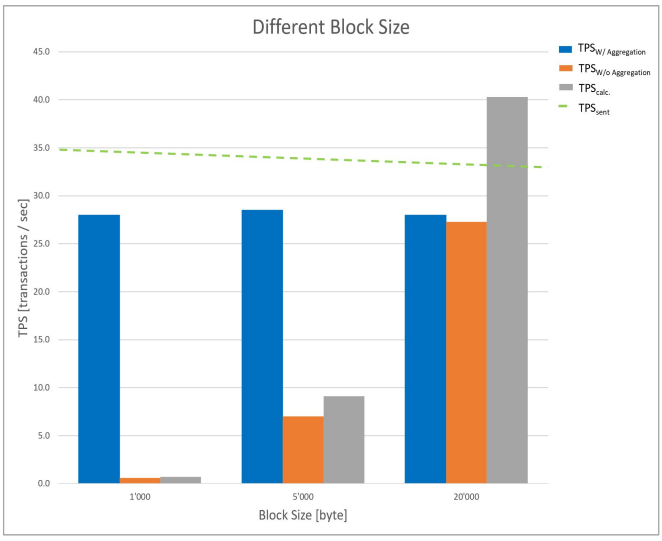

Fig. 8: Different block sizes and its influence on the *TPS* with and without aggregation

Figure 8 does visualize the output of tables II and III. The scalability improvement is clearly visible. One can say, that with transaction aggregation, the block size is not a limiting factor anymore if there are either same senders or same receivers in the transactions. This is visible in figure 8, because the blue bars are nearly constant, whereas the orange ones, standing for test runs without aggregation, are increasing with bigger block sizes. The grey bars do visualize the $TPS_{calc.}$. Furthermore it is surprising, that the orange bar is not higher in the test run with a block size of 20'000 byte. The block's size is not limiting the *TPS* anymore, since the $TPS_{calc.}$ is higher as the TPS_{sent}. This may indicate, that the connection issues described in section **??** and **??** are limiting the network throughput. Unfortunately, this assumption can neither be confirmed nor rejected because there is no fix for this problem yet. The TPS_{sent} is indicated by the green line.

The shrinking of the TPS_{sent} in figure 8 is not related to the block size, as it may could be assumed. Sending the transactions, even with a fixed interval, is still not always equally fast. It is related to the timespan in which a client receives an acknowledgment from a miner after sending the transaction. This timespan can slightly differ, and thus, it can have a bigger influence when sending a lot of transactions.

In the test runs with a block size of 1'000 byte, the version with transaction aggregation can handle that much more transactions because about 10 transactions fit into one block. These 10 transactions can be aggregation transactions aggregating way more transactions and therefore increasing the *TPS*. Here point two from the listing above does have an influence because 19 different wallets are in the network. When aggregating these transactions perfectly, it would result in 19 *AggTx* transactions. This would overflow the block size by nine transactions. If transactions from one sender cannot be validated in block *n*, there will be all these transactions plus the newly received ones for block *n + 1*. Thus, during the preparation of block *n + 1*, this specific sender will have more transactions than a sender whose transactions get validated in block *n* and therefore all its transactions get validated then.

The global distance of the network is not influencing the performance a lot since the ping-latency between the used virtual machines was usually below half a second during the test runs.

The block size is not limiting the BAZO version with transaction aggregation up to the point, where only distinct senders and receivers are sending and receiving the transactions. Therefore, in regards to the block size, transaction aggregation does increase the *TPS* especially for small blocks and thus it looks very promising.

VII. Different Block Intervals

Here the comparison and evaluation between different block intervals, with a given number of transactions sent to the network, is measured and listed.

Table IV does show the *TPS* rates for test cases with transaction aggregation enabled whereas table V shows the test cases without transaction aggregation. In both tables, the block interval has unit *seconds*, and the values belonging to transactions have unit *transactions* or *transactions per second*. In table IV the fraction $\frac{TPS}{TPS_{\text{calc.}}}$ can be seen as improvement factor in contrast to the theoretical maximum and in table V as degree of utilization.

The *ABI* (actual block interval) is an indicator if the blockchain does validate blocks in the user defined interval. It has unit *seconds* and is the average timespan between two blocks. Since the validation speed is set with the help of the *target*, it is not exactly the set interval [**?**]. This *target* is adapted every *n* blocks, whereas *n* is user defined.

The test runs with different block intervals look similar to the test runs with various block sizes. The *TPS* can be increased when transaction aggregation is used. Especially because the block interval and the *TPS* without aggregation are behaving inversely proportional, these test runs show the advantages nicely. The relationship between the *TPS* and the block interval is inversely proportional because, with a higher block interval, fewer blocks can be validated, and thus, less transaction as well.
The block size, on the other side, is related proportional to the *TPS*, because bigger blocks can handle more transactions. This results in a higher *TPS* value.
Consequently, the version without aggregation can theoretically handle the most transactions with a tiny block interval and huge blocks.

TABLE IV: Table of different block intervals influencing the *TPS* with transaction aggregation enabled.

Block interval		Transactions						
Defined	ABI	#sent	#validated	TPS_{sent}	TPS	TPS_{min}	TPS_{max}	$\frac{TPS}{TPS_{calc.}}$
15	18.8	181'933	181'933	33.3	28.5	20.9	33.2	3.1
60	66.8	190'000	190'000	34.8	34.4	33.9	34.7	15.3
120	118.4	181'278	181'278	33.8	32.9	30.2	33.4	28.5

TABLE V: Table of different block intervals influencing the *TPS* with transaction aggregation disabled.

Block interval		Transactions						
Defined	ABI	#sent	#validated	TPS_{sent}	TPS	TPS_{min}	TPS_{max}	$\frac{TPS}{TPS_{calc.}}$
15	14.3	74'960	74'834	34.7	7.0	7.0	7.0	0.8
60	63.5	78'375	78'375	36.3	2.0	2.0	2.0	0.9
120	111.4	18'000	18'000	34.4	1.1	1.1	1.1	1

Similar to the different block sizes, the block interval is not a *TPS* limiting factor anymore. Furthermore, transaction aggregation also allows validating more transactions per second, as already stated before.

During one test run, with a block interval of 60 seconds, the *TPS* is close to the TPS_{sent}. In this test run, due to certain unknown circumstances, the TypeID issue (section **??**) was not influencing so much as during other test runs.

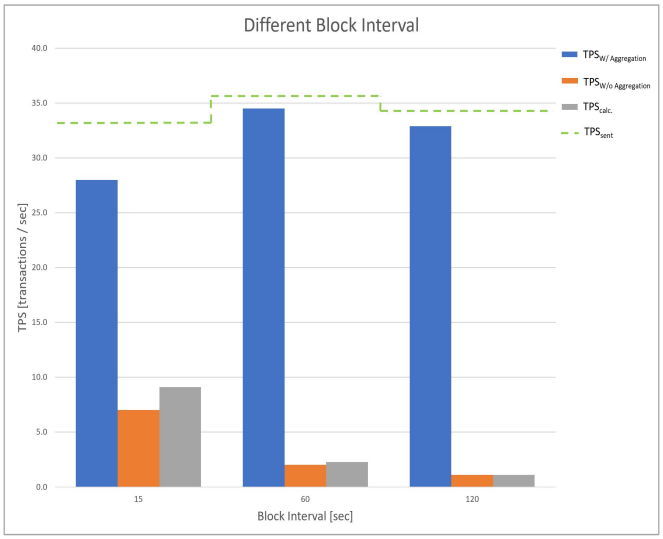

Fig. 9: Different block intervals and its influence on the *TPS* with and without aggregation

In figure 9 the differences of the TPS_{sent}, which is indicated with the green line, were relatively big and thus it has different heights. The differences are again caused by small differences between sending a transaction and receiving the acknowledgment from a miner.

The difference between the *TPS* with aggregation (blue bars in cart 9) and without (orange bars) is even bigger here, since the block interval and the *TPS* are inversely proportional.

It is not appropriate, to say, that with a higher block interval, the *TPS* can be increased for BAZO with aggregation enabled. Theoretically, the different block intervals should not have an effect on the *TPS* up to a certain number of different senders and or receivers. However, the differences can be founded on the TypeID issue again. This, because with the two higher block intervals, four to eight times fewer blocks have to be sent through the network compared to the 15 seconds interval. Therefore, the miners probably disconnect less often and they have more time to fetch transactions or blocks, which they never received.

Similar to the test with different sizes of blocks, the global distribution of miners and clients did not have a huge influence. The latency for a ping-test was usually around half a second.

Summarizing these outcomes results in the same perception as in section VI. Transaction aggregation does definitely increase the transactions possible to handle per second. Especially for a large block interval, similar to small defined block sizes, summarizing transactions increases the throughput enormously. Since the relation between the block interval and the *TPS* is inversely proportional, the difference in regards to the *TPS* with and without aggregation is even bigger, as with different block sizes.

VIII. Blockchain's Overall Size

In this section, the blockchain's overall size is analyzed and the findings, based on the test scenario defined in section **??**, are discussed.

As it is visible in graph 10, the blockchain size can be reduced with aggregation and even more with aggregation and emptying of blocks, once all transactions are aggregated. The difference between only aggregating and aggregating with emptying is not extraordinary big, because, in this test case only three different transactions are written into a block with aggregation. When emptying a block, the block's size gets reduced only around three times the size of a transaction hash. The more different transactions are listed in a block, the bigger this difference will be.

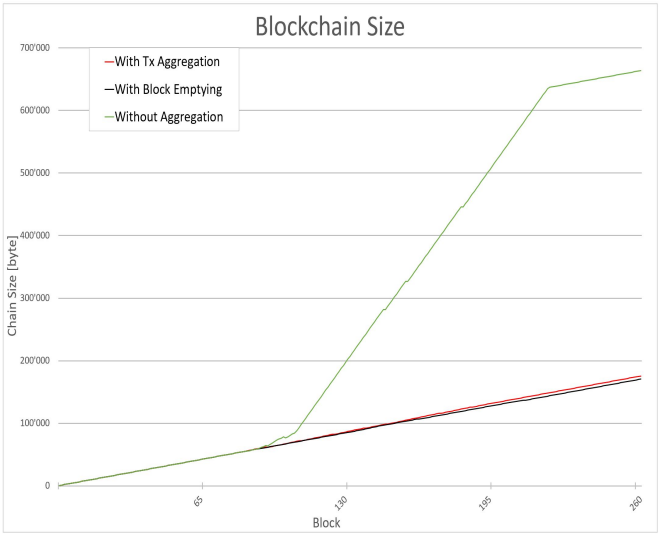

Fig. 10: Differences Regarding The size of The Blockchain With Aggregation, With Aggregation and Emptying of Blocks and Without Aggregation.

When a BAZO version with transaction aggregation (red or black line in figure 10) is compared to the version without (green line), a huge difference is visible. The difference is this big, because with transactions aggregation enabled, around three transactions are validated in each block, whereas without aggregation, roughly 135 transactions get aggregated in each block. These 135 transactions are also the maximum capacity of a block with the defined size of 5'000 byte. The two major kinks are caused by the start and end of sending transactions, whereas the smaller ones are caused by rollbacks. This graph shows the possibility of having a smaller overall blockchain size with transaction aggregation and emptying of blocks.

However, since self-contained proofs are not implemented in this aggregation technique, a new joining miner still has to fetch all transactions, no matter if BAZO is aggregating or not. And because aggregating transactions brings an extra transaction each time some transactions get aggregated, a new joining miner has to fetch even more transactions actually. This will change, once self-contained proofs are implemented. Then only the transactions since, *e.g.*, the last *epoch block* needs to be fetched from the network. This reduces the amount of data which needs to be fetched.

IX. Benefits Of Transaction Aggregation

Transaction aggregation does definitely allow more transactions in one block. Thus the overall throughput increases and at the same time the block size and the overall chain size can be kept small. Furthermore, the block interval can be enlarged, which reduces network traffic. However, transaction aggregation does help the most, if there are similar senders or receivers of transactions. It performs better, the more transactions with the same sender or receiver are sent. As BAZO is becoming an IoT blockchain, where many IoT nodes send their transactions to one receiver, this aggregation approach helps in scaling the blockchain.

As stated above, the implementation presented in this thesis does not help if there is no overlapping in terms of sender or receiver. Therefore, another aggregation technique should be used. A different possibility would be: Aggregate transactions *(A →B : 5)* and *(B →A : 20)* as one transaction *(B →A : 15)*. Here the only the final amount and the direction of the transaction gets written into the blockchain. This idea is kind of similar to *state-channels* proposed in section **??**.

Although transaction aggregation already works well here, its full power may only be visible once sharding and self-contained profs are implemented and combined with this technique.

X. Obstacles Of Transaction Aggregation

The intention behind double linking the blockchain is emptying all blocks once they are secure enough. A block is secure enough when it is accepted by the majority of miners, and thus, will not be included in rollbacks anymore. The emptying helps to save storage, as visible in section VIII. However, the emptying of validated blocks is kind of a contradiction against the core concept of a blockchain, where secure blocks are immutable and cannot be changed again. Thus, some impediments occur, especially when a new miner joins the network and wants to start mining.

A. Join As A New Miner & Order Of Transactions

When aggregating transactions in a historic manner, as it is described in section IV-2, various problems and difficulties can occur. They are described with the help of figure 11.

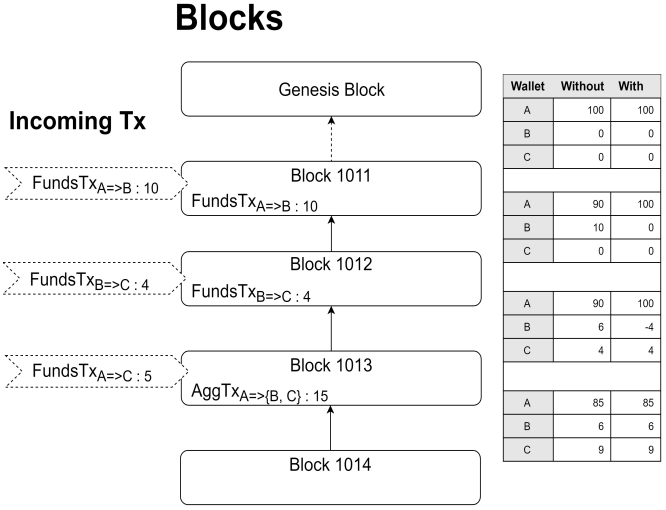

Fig. 11: Transaction aggregation and the balance.

In figure 11 three FundsTx are incoming to the blockchain and get validated in in blocks *1011*, *1012* and *1013*. As it is visible, the third transaction ($FundsTx_{A=¿C : 5}$) can be aggregated with the first transaction ($FundsTx_{A=¿B : 10}$), because of the similar sender. This results in the $AggTx_{A=¿\{B, C\} : 15}$ in block *1013*, and the removing of $FundsTx_{A=¿B : 10}$ in block *1011*.

The table on the right side shows the balances for the three wallets *A*, *B* and *C* with and without aggregation, before, between and after the three blocks are validated.

As it is now visible in the table, the balances for *A* and *B* are not the same when aggregating the transaction as when not aggregating them. This can lead to problems, especially when restarting or joining the network after transactions are already sent. Since BAZO fetches all blocks from the last validated one to the genesis block first and afterward validates them in the correct order, moving and aggregating transactions is problematic.

As example, when the transactions $FundsTx_{A=¿B : 10}$ and $FundsTx_{A=¿C : 5}$ get aggregated to $AggTx_{A=¿\{B, C\} : 15}$ and thus transaction $FundsTx_{A=¿B : 10}$ moves from block *1011* to *1013*, *B* does not have enough funds for transaction $FundsTx_{B=¿C : 4}$ at the point of validating block *1012*. This is visible in the middle two sub-tables where the balance of *B* is not the same with and without aggregation. When a new miner is joining the network, which uses transaction aggregation, and the $FundsTx_{A=¿B : 10}$ is not in Block *1011* but in *1013*, this new joining miner is not able to validate block *1012*, because *B* does not have enough funds.

One Way of eluding this is a credit-like behavior on startup. This concept allows a wallet to have a negative balance during the startup process. At the end, similar to the fourth small table in figure 11, the balances with aggregation enabled are the same as when validating each transaction without aggregation. As long as all transactions are validated, the order of validation does not play a substantial role. At a participation of a miner, this new miner only validates transactions which are already validated from other miners in the network. If the bootstrap miner tries to send invalid transactions to the new miner, the currently joining miner will find them, either by invalid block hashes, or when other miners are refusing its mined blocks later. Consequently, with this credit like behavior during the participation, it is possible to validate block *1012* even with aggregation and join the network.

Joining as a new miner, when blocks are already emptied, is problematic since the nonce of a block is calculated with the help of the wallets' balances. Here, similar problems as in the previous subsection can occur, and blocks are not validated because the nonce is incorrect at this point. This should also be

possible to prevent, when not checking the nonce on startup. It is also possible to argue, that these blocks are validated in the network already and therefore secure.

XI. Conclusions

Blockchains have been used to create a decentralized ecosystem for peer to peer transactions between non-trusting peers. Trust in BC is provided with consensus mechanisms that mainly give an equal sense and view of the whole system to all of it's users. To tackle the problems experienced with PoW-based consensus mechanisms, this work proposed the design and implementation of a PoS blockchain called BAZO. BAZO defines a semi-synchronous system, such that every node has its own local time. If a node attempts to speed up his hashing power, the system detects the malicious node and the suggested blocks are being rejected. BAZO determines the waiting time for validators to control the fair distribution of validator selection in the system. Without a minimum waiting time, a malicious node can easily reelect itself. In BAZO each validator owns a local timer and the whole system protects validators from diverging. Forks are controlled in this PoS based BC and scalability issues of PoW based BCs are addressed using transaction aggregation. To increase the randomness, local seeds are replaced by RSA signatures at every Block Height in order to determine the next eligible validator. Another key advantage of this PoS protocol is that a validator is always elected unless all validators are offline at the same time. It has been proved that along with high transaction rate, using BAZO is an environmentally friendly PoS-based BC, and proposes a higher degree of security than other implementations of PoW or PoS based BC such as in PPCoin and Tendermint. BAZO implements a chain-based PoS protocol to leverage the simplicity of this type of protocols. For future work, it is planned to evaluate and enhance the scalability of BAZO by employing proper methods to handle high rates of data streams, while assuring the fairness and randomness of the PoS consensus mechanism along with security aspects.

References

[1] ***Bitcoin Energy Consumption Index***. https://digiconomist.net/bitcoin-energy-consumption, [Last visit Dec 15, 2018].

[2] ***Bitcoin, Proof of Work***. https://en.bitcoin.it/wiki/Proof_of_work, [Last visit June 6, 2018].

[3] Thomas Bocek Burkhard Stiller:. ***Smart Contracts - Blockchains in the Wings***, pp 169–184. Springer, Tiergartenstr. 17, 69121 Heidelberg, Germany, Jan 2017.

[4] Vitalik Buterin. ***Proof of Stake FAQs***. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs, [Last visit Dec 15, 2018].

[5] Daniel Cawrey. ***Are 51% Attacks a Real Threat to Bitcoin?*** https://www.coindesk.com/51-attacks-real-threat-bitcoin, [Last visit Dec 15, 2018].

[6] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. ***On The Security and Performance of Proof of Work Blockchains***. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp 3–16. ACM, 2016.

[7] A.Mizrahi I.Bentov, A.Gabizon. ***Cryptocurrencies Without Proof of Work***. https://assets.ctfassets.net/sdlntm3tthp6/resource-asset-r351/5843a7c05e64302636f9139dd6438943/df8fc572-b128-44fc-8b16-0d06b87d5109.pdf, [Last visit Dec 15, 2018].

[8] J.Tuwiner. ***Peer-to-Peer Crypto-Currency with Proof-of-Stake***. https://www.buybitcoinworldwide.com/mining/hardware/antminer-s9/, [Last visit Dec 15, 2018].

[9] I.Eyal A.Gencer A.Juels A.Kosba A.Miller P.Saxena E.Shi S.Gün E.Sirer D.Song and R.Wattenhofer k.Croman, C.Decker. ***On Scaling Decentralized Blockchains***. In: Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff (Editors), *Financial Cryptography and Data Security*, pp 106–125. Springer Berlin Heidelberg, 2016.

[10] Daniel Kraft:. ***Difficulty Control for Blockchain-based Consensus Systems***, Mar 2016.

[11] Jae Kwon. ***Vitalik Buterin: Proof of Stake FAQ***. https://tendermint.com/static/docs/tendermint.pdf, [Last visit Dec 15, 2018].

[12] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. ***Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction***. Princeton University Press, 2016.

[13] Sina Rafati Niya and Burkhard Stiller. Bazo: Design and evaluation of a proof-of-stake (pos) based blockchain. Technical report, Zürich, Switzerland, May 2019.

[14] Nxt:. ***What is Nxt?*** https://nxtplatform.org/what-is-nxt/, Feb 2017.

[15] R.Blum:. ***Cryptographic Sortition for Proof of Stake in Bazo***. http://ifsoftware.ch/SemProgAnTr/files/Cryptographic_Sortition_for_PoS_in_Bazo_Draft.pdf, April 2018.

[16] S.Bachmann. ***Proof of Stake for Bazo***. https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Simon-Bachmann.pdf, Jan 2018.

[17] Livio Sgier. ***Bazo - A Cryptocurrency from Scratch***. https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Livio-Sgier.pdf, Aug 2017.

[18] S.Nadal S.King. ***Review Of Cryptonote White Paper***. https://peercoin.net/assets/paper/peercoin-paper.pdf, [Last visit Dec 15, 2018].

[19] Jordan Tuwiner. ***Bitcoin Cloud Mining***. https://www.buybitcoinworldwide.com/mining/cloud-mining/, [Last visit Dec 15, 2018].

[20] S.Micali G.Vlachos N.Zeldovich Y.Gilad, R.Hemo. ***Algorand: Scaling Byzantine Agreements for Cryptocurrencies***. https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf, [Last visit Dec 15, 2018].