Testing Security for Proof of Stake Implementations

Leo Cao Nathan Yu Louis Hu

Abstract

Security is a fundamental issue in blockchains, as it ensures the integrity of the blockchain, the accuracy of transactions, and the trust of users. This paper presents the design and implementation of our Proof of Stake (PoS) blockchain simulation to test different security strategies against various attacks. PoS is a consensus mechanism used in blockchain networks as an alternative to the traditional Proof of Work (PoW) mechanism. In PoS, validators are selected to validate blocks and secure the network based on the amount of cryptocurrency they hold and "stake" as collateral. A common malicious behavior is double spending, where the same funds are used for multiple different transactions. Attacks on the blockchain that give the dishonest user opportunities to conduct fraudulent activities like double spending compromise the accuracy of transactions.

Our simulation supports a few different methods of enhancing blockchain security, including the standard PoS consensus, slashing of misbehaving validators, and reputation based PoS. There are a couple of attacks that can be simulated: network partition and balance attack. We report the measurements about how the different strategies for blockchain security perform against our simulated attacks.

1 Introduction

Blockchain technology has gained significant attention in recent years as a secure and transparent means of recording transactions. In this paper, we present the design and implementation of our Proof of Stake blockchain simulation, aimed at testing various security strategies against different types of attacks. PoS is a consensus mechanism used in blockchain networks as an alternative to the traditional Proof of Work mechanism, where validators are selected to validate blocks and secure the network based on the amount of stake they hold in the system. One common malicious behavior in blockchains is double spending, where the same funds are used for multiple transactions, compromising the accuracy of transactions.

Our simulation supports several methods of enhancing blockchain security, including the standard PoS consensus, slashing of misbehaving validators, and reputation-based PoS. We also simulate attacks such as network partition and balance attacks to measure the performance of different security strategies against these simulated attacks.

The implementation of the simulation is developed in Golang, with a centralized global coordinator process and each machine in the blockchain represented as a goroutine. The simulation aims to measure the performance of the different blockchain security strategies against attacks such as network partition and balance attacks. For the attacks in our simulation, we chose to focus on long-range and short-range forks caused by malicious nodes, as they are feasible to implement and allow us to test the effectiveness of our slashing, rewarding, and election systems. We believe that our simulation will provide valuable insights into the performance of various security strategies against simulated attacks in different PoS blockchain implementations. Overall, this paper's findings could help enhance the security of PoS blockchains and improve their resilience against malicious activities.

2 Background & motivation

2.1 Proof-of-work

Bitcoin [12], the predominant cryptocurrency, uses proof-of-work to achieve consensus on a set of approved transactions, this approach is often referred to as the "Nakamoto consensus." Proof-of-work requires miners to perform complex computational tasks to validate transactions and add new blocks to the blockchain. The first miner to solve the computational puzzle and validate the transaction is awarded with Bitcoin. Once this new block is broadcasted to all the other receivers and approved, then the block can be appended onto the blockchain. This process is both energy- and time-intensive, as miners use powerful hardware to perform the computations, and these computations take a long time.

The PoW algorithm was designed to solve the problem of double-spending, which is a significant challenge in decentralized payment systems. Double-spending occurs when a user spends the same cryptocurrency twice, which can lead to fraud and undermine the integrity of the payment system.

PoW has several advantages. One of the main advantages is that it ensures that the blockchain network is secure and decentralized. The computational tasks required to validate transactions are designed to be difficult and time-consuming, making it difficult for malicious actors to gain control of the blockchain network. Additionally, PoW ensures that miners are incentivized to act honestly, as they can earn rewards in the form of cryptocurrency.

However, PoW also has several drawbacks. One of the main drawbacks is that it is energy-intensive, as miners use powerful hardware to perform the computational tasks required to validate transactions. This has led to concerns about the environmental impact of PoW and the sustainability of the blockchain network. Secondly, Bitcoin has to balance the length of time to compute a new block with the possiblity of wasted work [12], and set parameters to generate a new block every 10 minutes on average.Moreover, due to the possibility of forks, users usually have to wait for the blockchain to grow by at least six blocks before considering their transactions confirmed [3]. This means transactions in Bitcoin can take hours before being confirmed. This has led to concerns over the feasiblity of Bitcoin as a reliable form of currency. Additionally, PoW can lead to a concentration of mining power in the hands of a few, as miners with more powerful hardware have an advantage over others. This leads to centralization which is the opposite of what Bitcoin wants which is a decentralized network.

Despite its drawbacks, Proof-of-work remains a popular consensus mechanism used by many blockchain networks. However, there has been increasing interest in alternative consensus mechanisms, such as Proof of Stake, which is more energy-efficient and democratic. Our project focuses on Proof of Stake, as we believe it has the potential to address the challenges associated with Proof-of-work and provide a more efficient, fair, and secure consensus mechanism for blockchain networks.

2.2 Proof-of-stake

Proof of Stake (PoS), on the other hand, is a consensus mechanism that was first introduced by [9] and now used by Ethereum and other cryptocurrencies. Proof of Stake runs on the conecpt of giving decision-making power regarding who gets to append to the blockchain to nodes who possess coins, or otherwise known as "stake", within the system [2]. The idea behind Proof of Stake is that nodes who hold stake in the system are well-suited to maintain security, since their stake will diminish in value if the security of the system were to be corrupted. Therefore, similar to Bitcoin, an individual stakeholder who posseses $p\%$ of the stake in circulation will have a $p\%$ chance of being able to create the block and append it to the blockchain.

Proof-of-stake has numerous advantages over proof-of-work. First, the most apparent improvement is the lower energy usage. There is no longer a need to use lots of energy on proof-of-work computation. Second, PoS is more democratic due to lower barriers of entry and reduced hardware requirements–any laptop can join the PoS consensus protocol. This is in contrast to proof-of-work, where miners with more powerful hardware have an advantage over others, resulting in a concentration of mining power in the hands of a few. Lastly, PoS is more secure because it makes it difficult for malicious actors to gain control of the blockchain network.

Several blockchain platforms have adopted PoS as their consensus mechanism, including Ethereum, the second-largest cryptocurrency by market capitalization. The Merge was executed on September 15, 2022, and completed Ethereum's transition to proof-of-stake consensus, officially depricating proof-of-work and reducing energy consumption by 99.95% [8].

Proof-of-stake's disadvantages are directly related to its advantages. One of the main challenges is ensuring that the consensus mechanism is secure and stable. PoS is susceptible to attacks that can cause instability in the system, such as long-range attacks including balance attacks, and short-range attacks including Nothing-at-stake attack. Therefore, it is essential to design PoS implementations that are secure, stable, and scalable, in order to create a reliable blockchain for one's cryptocurrency

3 Implementation

We have considered several implementations for our simulator, such as which PoS blockchains to use and which types of attacks would best reflect the strengths and weaknesses of each blockchain. Our aim was to select attacks that could be feasibly implemented while accurately capturing the nuances of each blockchain's security features.

Our simulator was developed in Golang as we wanted to take advantage of existing blockchain project structures provided by sources such as the following blog post provided by coralhealth [10]. Although we initially used online examples such as this post as a means to start our project and learn Golang, we soon deviated and made the simulator our own.

The organization of the simulator consists of a centralized, global coordinator process with each machine in the blockchain being represented as a goroutine with the corresponding state. Goroutines are basically just threads which are very simple to use in Golang. To run our simulation we change our desired parameters in the run function in main.go which starts our centralized process. This process then accepts socket connections on the same machine which will represent new users or validators. Based on the inputs for initialization, these machines will be instantiated as structs in their own separate process where they may then accept requests to operate in the blockchain or create transactions. We chose this type of design without simulating heavy network communication because we wanted our simulation to only factor in the design of the blockchain not the attributes of the network when measuring blockchain performance to various attacks.

Initially while reading through multiple evolutions of standard proof of stake, we decided on delegated proof of stake and reputation based delegated proof of stake, both of which were supposed to be improvements on electing honest nodes at a higher rate while being more efficient in computation. However, while implementing these blockchains we realized that delegated and reputation based delegated would be too similar in the context of our simulation so we decided on standard proof of stake, reputation based delegated proof of stake, and a variation of the standard proof of stake with slashing. We believe that these three implementations would show how attacks perform in response to delegates, rewards, and punishments of validator behavior.

For our attacks, we had initially looked into long ranged balance attacks, denial of service attacks, and a simple 51% attack. However after reconsidering our metrics we saw that denial of service attacks did not highlight advantages of the different blockchain implementations and that the 51% attack was too simple to discern a difference in metrics as well. We decided on taking advantage of long range and short range forks caused by malicious nodes as they were feasible to implement and tested our slashing, rewarding, and election systems.

3.1 Proof of Stake Blockchains

Standard Proof of Stake with and without Slashing

Our standard proof of stake blockchain is implemented in a way where the centralized coordinator contains the last fully agreed blockchain, represented as an array of blocks. The validators in the network will each have their own views of the chain that they add blocks to and try to reach a consensus on which chain should be finalized. The overall flow of blockchain operation consists of incrementing through fixed time slots. During each time slot at most one block is supposed to be proposed by a proposer then validated by a committee of validators to be appended to the blockchain. All the while transactions from users are asynchronously generated as messaged to validators' transaction pools. We randomized the amount of starting stake for each validator and included rewards for validating certain transactions.

Validators

Validators are nodes/machines in the blockchain network that are responsible for actions such as block creation, validation, and voting on aspects of the current state of the blockchain. In our implementation, a socket connection can be made with the main process which will spawn a struct in a separate goroutine thread to represent the newly instantiated validator. Validators contain their own views of the blockchain, channels for communication with the coordinator, address, stake, transaction pools, reputation, and metric variables such as the number of successful proposals and whether or not they are malicious. Validators generate blocks and validate blocks and transactions. Our implementation has each validator thread spawn a new thread for listening for messages for action. These messages include requests to validate blocks, updating transaction pools, and voting in delegate elections. All such communication is done between threads through Golang channels.

Users

Users are similar to networks in their instantiation through socket connections and later in memory structs in a different thread. Users mainly serve to asynchronously generate transactions to be broadcast to validators which can choose to accept them into their transaction pool and include them in future blocks. Users contain cryptographic public and private keys used for validating and signing transactions along with balances. Validators will check whether the transaction was already spent, the user has sufficient funds, and whether or not the public key can be used to verify the transaction all before the validator includes the transaction in its pool.

Validation and Block Proposal

In each time slot, a committee of validators is chosen pseudo-randomly, weighted by stake without replacement to serve on the validation committee. A block proposer will be chosen from this committee with the weighted stake process. After this block proposer is chosen, it generates a block from the transactions in its pool and attempts to broadcast the block to the committee for validation. The committee will then validate and vote on whether or not to add the block and the majority decision will be broadcast to all blocks to either add or ignore the block. To validate a block, a validator must verify the recomputed hash is equal to the original and that the old block is the previous block in a validator's blockchain.

Finality

To ensure that our finality-based attacks function correctly, we implement finality through the existence of finalized parts of the blockchain and recent parts of the blockchain that remain to be finalized. During this temporary window which is every fixed number of time slots, forks and differing views of the blockchain are possible either through malicious forks or network partitions which all could threaten the blockchain. To reach consensus after every fixed number of time slots, the validators must consolidate all differing views of the blockchain and agree that the longest blockchain is correct and that the chain will be finalized up to that point, making it very difficult to overwrite any existing transactions or block data.

Slashing

For our slashing based proof of stake algorithm, we slash a validator's stake for any perceived malicious actions. These actions include validating invalid blocks, proposing invalid blocks, and creating forks. In this way the blockchain can make sure that malicious actors do not come into positions of power where they may allow attacks to occur once again. Our value for slashing penalized proposing invalid blocks and creating forks by slashing stake down to 20% while we were more forgiving for possible confusion in validating invalid blocks by only slashing down 80% of original stake.

Delegated Proof of Stake

Delegated proof of stake blockchains offer an improvement on the traditional proof of stake by involving fewer nodes in the overall process, instead relying on the trust and reputation of a handful of delegates to perform proposal, validation, and other common administrative actions. In this way delegated proof of stake trades a degree of decentralization for faster processing and supposedly more reputable blocks and operation.

Delegates

In our implementation, delegates are chosen pseudo-randomly much like the validation committee of the previous two blockchains, except instead of weighting by stake, delegates are chosen weighted by reputation. This is an abstraction of the voting process that happens in traditional delegated proof of stake blockchains where validators vote for delegates based on a variety of different, sometimes subjective qualities. We abstracted this voting system out because it was unnecessary to include such attributes in our reputation score. The size of the delegate committee is configurable, but we decided to make it a fraction of the size of the validation committee in order to show the contrasts in centralization and efficiency. Instead of new validators taking new roles every time slot, instead delegates are elected every election term which is much longer at every 5 time slots. Delegates then take turns in a sequential order in proposing blocks while the rest of the delegates validate and broadcast valid blocks. We see that this implementation is prone to malicious delegates staying for too long in the delegate committee along with the dangers of centralization if honest nodes cannot establish their reputation in time.

Reputation

Our reputation system is heavily inspired from Hu et al. 2020 [1]. Reputation is much like slashing where the same malicious actions mentioned in slashing will slash reputation by the same amount. Where reputation differs is its reward system. For proposing a valid block, actively voting, and actively validating, a validator will see its reputation increase. What this means is if honest delegates perform their duties normally their reputation will increase at a faster rate than the non-delegates' reputation increases just for actively voting. This usually ensures that honest delegates will stay on the committee indefinitely until malice is detected. We start each validator with the same base reputation score of 5 which can grow up to 100 which is the max.

3.2 Proof-of-Stake Attacks

Network Partition Attack

The first class of attacks on proof-of-stake blockchains is short range attacks. In a short range attacks, the attack is only on a small number of blocks and targets at reorganizing blocks of a few days up to a few months. There are many attacks that fall under this category such as the Bribery attack [4] but we will look at the more general problem of the "nothing at stake" problem.

Nothing at Stake Problem

The nothing at stake problem is a theoretically security hole in proof-of-stake systems. The problem can occur at anytime there is a fork in the blockchain, either due to malicious action or when two honest validators accidentally propose blocks simultaneously [14].

Then whenever a fork is created, it is in a validator's incentive to try to make blocks on top of every chain. This is due to two reasons. One, there is no cost to mining so mining both chains does not impact a miner's balance. Two, since the miners do not get punished for producing blocks, mining both of the forks ensures that the miner will get their reward no matter which fork wins [14]. We can see this in play in the two figures 1 and 2 below [5].

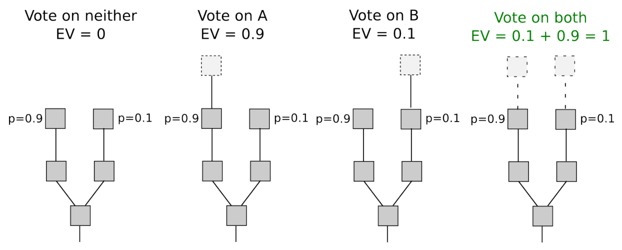


Figure 1: Nothing at Stake Problem in Proof of Stake

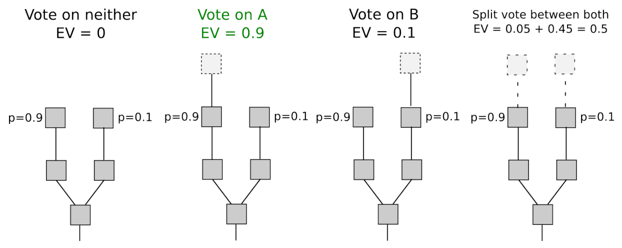


Figure 2: Nothing at Stake Problem in Proof of Work

This could lead to double-spend attacks being more feasible. An attacker can maliciously create a fork in the blockchain one block before they spent coins. Then the attacker could mine only their fork while all of the other miners act in their best self-interest and mine both forks, then the attacker's fork would become the longest chain and could then be eventually finalized, erasing the attacker's transaction [14]. This is a easily created security vulnerability in a proof-of-stake system, and our simulation aims to show how easily a naive pure proof-of-stake system can be compromised due to the nothing at stake problem.

Research has shown that this can be solved using two strategies. The first involves penalizing the validator if they simultaneously create blocks on multiple chains, by means of including proof of misbehavior, such as two conflicting signed block headers, into the blockchain. Another strategy involves punishing validators for creating blocks on the wrong chain [5]. We opted to use the first strategy in one of our proof-of-stake implementations, and slash the validator's stake if they maliciously create a fork in the blockchain.

Network Partition Attack Overview

Inspired by the nothing at stake problem, in our network partition attack, a malicious validator will create a fork on purpose by proposing two blocks in the same time slot. The attacker then will split the validators in half such that one half of the validators get one block and the other half get the other block. Due to the incentives explained in the last section, the validators will vote to approve each block on each fork. Then the attacker will partition the validators such that each half can only see their side of the fork. As a result, whenever a new block is proposed by a validator, only their half will see it. This will then only cause the fork to grow larger and the inconsistency in ledger to be greater. And as mentioned before, the attacker's double-spend attacks become much more feasible.

Network Partition Attack Implementation

In our implementation of the network partition attack, before we start the attack, we will initialize the groups that each validator is in a 2d Array of Validator pointers *ForkedBlockchain*. We then start the attack whenever the attacker becomes the proposer. Then if the blockchain has not been forked yet, we will create two new blocks, *newBlockOne* and *newBlockTwo*. Then after we collect the votes of the validators on the committee or delegates, we have 3 cases to look at. First case, if the blockchain is already forked, then we only broadcast the new block to the proposer's group in *ForkedBlockchain*. So even if the proposer is evil, it will only propose a block and act normal. Second, if the blockchain is not forked yet and the proposer is evil. Then we will perform the network partition attack and maliciously create a fork. Once the fork is created, the individual nodes would have different views of the blockchain. We simulate this by only broadcasting one block to the first group in *ForkedBlockchain* and the other block to the second group. Now the blockchain is forked so we move to first case once we run *nextTimeSlot()* again. The last case is if the proposer is not evil and the blockchain is not forked. Then we just run the usual proof of stake protocol. In each case, we will update transactional amounts and reward proposer accordingly if the block is added to the blockchain.

Balance Attack

The next class of attacks on proof-of-stake blockchains is long range attacks. In a long range attack, an attacker that was involved in the genesis block has enough stake in the system to deceive honest nodes into believing a separate fork. They split the views of the honest nodes so that enough of the honest nodes believe in the false chain, and the attacker is able to build this false fork, indistinguishable from the honest chain. Later at an opportune time, they convince the network to switch over to the malicious chain. The long range attack that we decided to implement and simulate is a balance attack [7].

Balance Attack Overview

In a balance attack, a malicious validator tries to create multiple forks of the blockchain by creating multiple blocks at the same time with different transactions, rather than following the consensus rules and creating only one block at a time. The attacker does this by splitting their stake into multiple smaller stakes and using each of them to create blocks simultaneously on different branches of the blockchain. This creates a situation where the attacker has multiple branches of the blockchain that they are trying to extend simultaneously. In a proof-of-stake (PoS) blockchain network validators are selected to create new blocks and validate transactions based on the amount of cryptocurrency they have staked. In a balancing attack, an attacker attempts to accumulate a large amount of cryptocurrency and split it between multiple accounts or addresses. The attacker then uses these accounts to stake on multiple chains simultaneously, making it more difficult for the network to reach consensus. By staking on multiple chains, the attacker can increase their chances of being selected as a proposer on each chain, and create a situation where the network is unable to agree on the validity of new blocks. This can cause a fork in the chain, where some nodes follow one set of blocks, while others follow another set, resulting in a split network.

The goal of the attacker in a balance attack is to double-spend their cryptocurrency, meaning they spend the same cryptocurrency twice by creating conflicting transactions on different branches of the blockchain. For example, the attacker might spend their cryptocurrency to buy a product from one merchant on one branch of the blockchain, and then simultaneously spend the same cryptocurrency to buy another product from a different merchant on another branch of the blockchain. The challenge for the blockchain network in this situation is to reach a consensus on which branch of the blockchain is

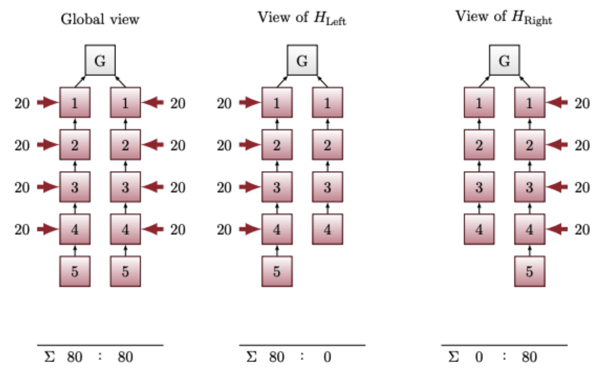


Figure 3: Validators in H_{left} and H_{right} see their side of the fork as the honest chain

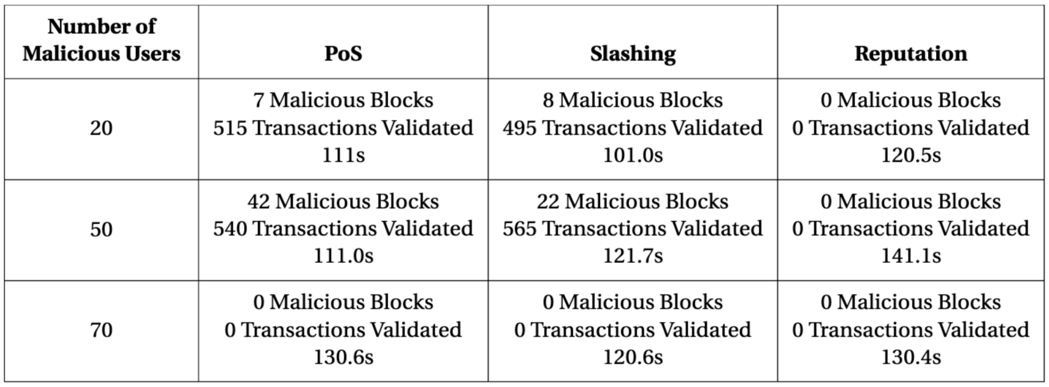
the valid one, as the attacker is actively creating multiple branches and trying to extend them. This can lead to a fork in the blockchain, where the network splits into multiple chains with different transaction histories, and it becomes difficult to determine which chain is the true version of the blockchain. To mitigate balance attacks, PoS blockchains often implement mechanisms such as checkpoints, where certain blocks are designated as "final" and cannot be reversed, and slashing, where validators who create multiple conflicting blocks are penalized by having their stakes slashed or confiscated. Additionally, PoS blockchains may also use various consensus algorithms or protocols, such as Delegated Proof of Stake, that introduce additional rules and mechanisms to prevent or mitigate balance attacks [13].

Balance Attack Implementation

Our implementation of a balance attack starts from a fork from the genesis block, which is slightly different from how a real balance attack would occur. Of course, it is very unlikely an attacker would be forking from the genesis block, instead it would wait until it is selected as the proposer, and propose two blocks to create a fork. We decided to ignore this proposal logic and essentially immediately choose a malicious validator to propose a block at genesis. This is for simplicity and the rest of the logic involved in a balance attack is not triggered until there is a fork anyways. Once the fork is created, the individual nodes would have different views of the blockchain. Since we have a central TCP server handling connections of different nodes, this view-splitting is handled by the central server setting nodes to a certain view when they connect. In a real balance attack, the attacker would have a view of the entire network (i.e. both sides of the fork). Since our validators are designed to have a view of a single blockchain, we simulated the attacker view by allowing the central server to broadcast which side of the fork is longer to the malicious nodes, so that they can vote in a way that maintains balance. The honest nodes see one side of the fork, and this is simply accounted for by our design, since each honest node has its own array of blocks that functions as its view of the network.

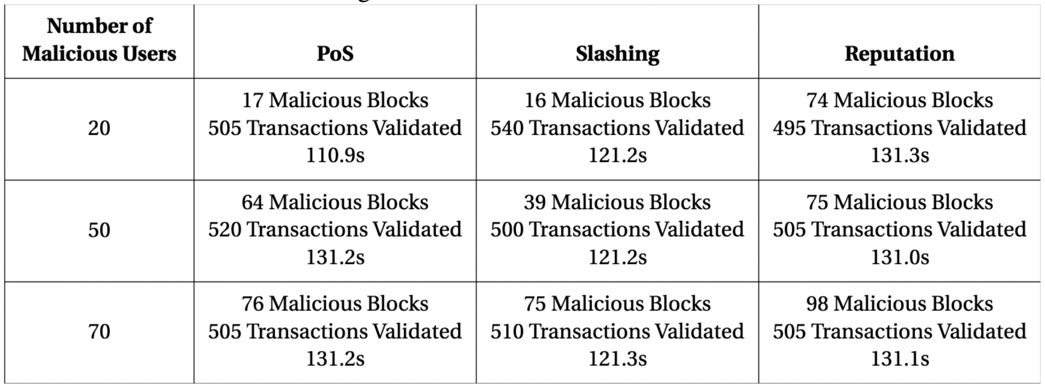
From there, the malicious validators play a role in trying to maintain the balance of the two forks in each round of proposal and voting, specifically in balance attack's "nextTimeSlot" function. Since the attacker/malicious validators can see both sides of the fork as explained above, they vote to balance the two sides if they are selected for the validation committee. Additionally, malicious nodes selected as the proposer would propose for the shorter side of the fork in hopes of that block being approved. During validation, when the proposed block is broadcast in a message to the members of the validation committee, the malicious validators approve the block if it is to be appended to the shorter side, and they would vote against the block if it is to be appended to the longer fork. More specifically, we implemented this by modifying the logic in our validator class. When a validator receives a message to vote on a block while we are running a balance attack, the validator calls a different function to check if the block is valid, and this function contains the logic necessary for malicious nodes to vote in the manner described above. For honest validators, they vote as if there was no attack happening, approving the proposed block if the new block's previous block hash is consistent with its own view of the previous block in the blockchain. After the validators vote, the central server processes all the votes and decides whether the block is appended in the same way as if there was no attack. As explained in our implementation of our blockchains, there is some randomization in the selection of the validation committee (or selection of delegates) based on the amount of stake the validator has in the system, so if the attacker does not have enough stake in the system, we see their votes be overpowered by the honest validators in the validation committee.

Our blockchain has a finality mechanism, where certain blocks are designated as "final" and cannot be reversed. We used a simple consensus strategy for finality, where we simply try to find the longest chain out of all the possible forks in the network, and if one chain is sufficiently longer than the others, we establish this chain as "certified" blockchain, and broadcast it to all the nodes in the network so that they are all on the same view of the blockchain. This checkpointing mechanism is common in blockchains that combats balance attacks specifically, as it reconciliates any forks given one chain is longer than the others. More specifically, in our implementation this longest chain consensus looks through all views of all the validators in the network, and if it finds a validator with a chain that is at least two blocks longer than the others, then it broadcasts that chain as the "certified" blockchain. So, this means the balance attack must maintain the two sides of the fork to be within one block of each other, to avoid consensus from being reached. Again, we see how well the attack is able to delay consensus relates back to how much control/stake the malicious validators have in the network. The specific effects



Number of Malicious Users	PoS	Slashing	Reputation
20	7 Malicious Blocks 515 Transactions Validated 111s	8 Malicious Blocks 495 Transactions Validated 101.0s	0 Malicious Blocks 0 Transactions Validated 120.5s
50	42 Malicious Blocks 540 Transactions Validated 111.0s	22 Malicious Blocks 565 Transactions Validated 121.7s	0 Malicious Blocks 0 Transactions Validated 141.1s
70	0 Malicious Blocks 0 Transactions Validated 130.6s	0 Malicious Blocks 0 Transactions Validated 120.6s	0 Malicious Blocks 0 Transactions Validated 130.4s

Figure 4: Results of Balance Attack



Number of Malicious Users	PoS	Slashing	Reputation
20	17 Malicious Blocks 505 Transactions Validated 110.9s	16 Malicious Blocks 540 Transactions Validated 121.2s	74 Malicious Blocks 495 Transactions Validated 131.3s
50	64 Malicious Blocks 520 Transactions Validated 131.2s	39 Malicious Blocks 500 Transactions Validated 121.2s	75 Malicious Blocks 505 Transactions Validated 131.0s
70	76 Malicious Blocks 505 Transactions Validated 131.2s	75 Malicious Blocks 510 Transactions Validated 121.3s	98 Malicious Blocks 505 Transactions Validated 131.1s

Figure 5: Results of Network Partition Attack

of different numbers of malicious validators will be discussed more in-depth in the results of our simulation.

4 Results

In this section we will be discussing and interpreting the results from running our simulation with our different configurations. Our main tests consisted of varying the number of malicious validators, the proof of stake implementation between standard, slashing, and reputation based blockchains, and varying the attack between network partition and balance attack. Parameters we kept constant were that our simulation would have 100 validators, 10 users, 20 committee members in PoS and slashing, 5 delegates in reputation PoS, and that we would record metrics after 100 blocks on the blockchain. We chose to vary number of malicious users to represent not a majority, 50/50, and majority. We chose for 20 committee as not all validators are included and it gives higher stake nodes more power and influence. We wanted to choose for delegate size to be 5 because it shows the increased centralization and benefits for reducing the number of active nodes.

Metrics for our simulation include number of malicious blocks approved, number of transactions validated, and time to validate 100 blocks.

For our simulation of balance attacks on our blockchains we see the following results in Figure 4. Key takeaways are that as the number of malicious validators increase, so does the number of malicious blocks validated which is to be expected. Except that this turns to 0 blocks and 0 transactions validated at certain points such as 70 malicious validators in PoS and Slashing. For Reputation no configuration with malicious nodes validated any blocks. This 0 blocks validated and 0 transactions validated shows how the balance attack effectively prevented any form of consensus or block approval by constantly balancing competing views, allowing for disruption of the network. We see that reputation delegated proof of stake is the most vulnerable here to this attack most likely due to the fact that malicious fork creating validators cannot be removed before a delegate election term meaning that they cannot be punished in time. The slashing blockchain performed the best allowing the fewest malicious blocks to be proposed with 50 malicious validators, however it was still not immune to the balance attack at 70 malicious validators.

For our simulation of network partition attacks on our blockchains we see the following results in Figure 5. Key takeaways are that as the number of malicious validators increase, so does the number of malicious blocks validated which is to be expected. Once again slashing seems to perform marginally better when malicious validators are only at 50, but once again once a majority of nodes are malicious it is equally susceptible as standard PoS. Along with this Reputation once again performs the worse in all metrics. Even with 20 malicious users, 74 malicious blocks are being approved out of the 100 measured. This speaks less about the design of the blockchain, but rather our parameters for the simulation. Because of the lengthened delegate terms combined with our reputation system, malicious nodes can stay in power a few terms after they have attacked. This allows them to wreak havoc even after being outed as a malicious node. Along with this because we only measured 100 block approvals, this could not have been a sufficient amount of time for reputation to entirely establish itself as punishments are not harsh enough to punish all malicious validators at once.

5 Experience

5.1 Teammate Responsibilities

Although we each worked on a separate section of the project, it was much more collaborative in nature where each of us would help where we could and try out different parts of the project. Louis worked on design and implementation of the short-range network partition attack. Leo worked on design and implementation of the long-range balance attack. Nathan worked on design and implementation of the blockchains and simulation.

5.2 Limitations

There are a few limitations in our simulation. One, each time we need to add a new implementation of an attack or variant of Proof of Stake, we have to edit the source code ourselves. Eventually, we would like this process to be decentralized so that a user can plug in their own variant and run their own attacks.

Another limitation is the lack of the global view of the blockchain. In our design, each validator stores their view of the blockchain individually. As a result, attacks like long-range attacks or nothing-at-stake attacks involve manipulating stake or generating forks from different blockchain states requires some workarounds to give the malicious user the global view of the network that they would have during the attack. Additionally, gathering measurements for these attacks is more difficult since it requires more effort to achieve a comprehensive understanding of the blockchain.

For our simulation we see that our results could gain credibility from running simulations with 0 malicious nodes to serve as a constant base. Along with this, our simulation doesn't incorporate natural network partitions or communications on a truly decentralized network which are further areas we would want to explore. For our metrics we had a hard time measuring double spending so we could not quantify the true amount of damage each blockchain was doing. This speaks to the point as well that many of the attacks we researched had different goals of capitalization vs destablilization which limited which attacks we wanted to focus on. Along with this our metric for the number of malicious blocks validated might be misleading because these could be perfectly valid blocks being proposed by a malicious validator who is not acting malicious at the moment.

5.3 Future Work

This paper explores the impact of network partition and balance attacks on committee-based proof of stake and delegated proof of stake. However, there is still a lot of work to be done in the future to continue improving our proof of stake attack simulator. One area of focus will be the implementation of new variants of proof of stake, including BFT-based PoS, Liquid PoS, Ouroboros [11], and Ethereum's Gasper protocol [6]. Additionally, new attacks, such as stake-bleeding attacks, should be developed to simulate against these new variants. It is also important to explore alternative countermeasures to attacks beyond the longest chain rule, such as checkpoints, key-evolving cryptography, Context-Aware Transactions, and Plenitude Rule [7]. These changes will require signifcant time and effort, as they are highly complex features. To faciliate this process, in the future, a higher-level, abstracted simulation where one can simply input their own version of proof of stake and run our attacks from there.

6 Conclusion

In conclusion, our research paper presented the design and implementation of a Proof of Stake (PoS) blockchain simulation to test different security strategies against various attacks. We evaluated the performance of different PoS implementations, including standard PoS, slashing of misbehaving validators, and reputation-based PoS, using simulated attacks such as network partition and balance attack. Our results showed that varying the number of malicious validators, the PoS implementation, and the type of attack had an impact on the security of the blockchain. Specifically, we observed that increasing the number of malicious validators compromised the accuracy of transactions, while different PoS implementations showed varying levels of resilience against attacks. Our findings highlight the importance of robust security measures in blockchain networks to maintain the integrity of the blockchain, accuracy of transactions, and trust of users. Further research can be done to explore additional security strategies and their effectiveness in different scenarios to enhance the security of blockchain networks.

7 Acknowledgements

We thank Danyang Zhuo for all the support and feedback throughout all the checkpoints.

References

[1] An improved delegated proof of stake consensus algorithm. *Procedia Computer Science*, 187:341–346, 2021. 2020 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI2020.

[2] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work, 2017.

[3] BitcoinWiki. Confirmation. https://en.bitcoin.it/wiki/Confirmation, 2018.

[4] Joseph Bonneau. Why buy when you can rent? bribery attacks on bitcoin-style consensus. In Kurt Rohloff, Jeremy Clark, Sarah Meiklejohn, Dan Wallach, Michael Brenner, and Peter Y.A. Ryan, editors, *Financial Cryptography and Data Security - International Workshops, FC 2016, BITCOIN, VOTING, and WAHC, Revised Selected Papers*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 19–26. Springer Verlag, 2016. International Workshops on Financial Cryptography and Data Security, FC 2016 and 3rd Workshop on Bitcoin and Blockchain Research, BITCOIN 2016, 1st Workshop on Advances in Secure Electronic Voting Schemes, VOTING 2016, and 4th Workshop on Encrypted Computing and Applied Homomorphic Cryptography, WAHC 2016 ; Conference date: 26-02-2016 Through 26-02-2016.

[5] Vitalik Buterin. Proof of stake faq. https://vitalik.ca/general/2017/12/31/pos_faq.html#what-is-the-nothing-at-stake-problem-and-how-can-it-be-fixed, 2017.

[6] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper, 2020.

[7] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. A survey on long-range attacks for proof of stake protocols. *IEEE Access*, 7:28712–28725, 2019.

[8] Ethereum. The merge. https://ethereum.org/en/roadmap/merge/, 2023.

[9] Bitcoin Forum. Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0, 2011.

[10] Coral Health. Code your own proof of stake blockchain in go! https://mycoralhealth.medium.com/code-your-own-proof-of-stake-blockchain-in-go-610cd99
2018.

[11] Florian Mellentin. Ouroboros: Cardano's proof-of-stake consensus protocol, 02 2021.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.

[13] Joachim Neu, Ertem Nusret Tas, and David Tse. Two more attacks on proof-of-stake ghost/ethereum. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, ConsensusDay '22, page 43–52, New York, NY, USA, 2022. Association for Computing Machinery.

[14] Julian Roberto. Understanding proof of stake: The nothing at stake theory. https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-the
Jun 2022.