FAIR AND EFFICIENT CONSENSUS PROTOCOLS FOR SECURE BLOCKCHAIN APPLICATIONS

by

Golam Dastoger Bashar

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Computing, Cybersecurity

Boise State University

December 2021

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the dissertation submitted by

Golam Dastoger Bashar

Dissertation Title: **Fair and Efficient Consensus Protocols for Secure BlockchainApplications**

Date of Final Oral Examination: 18 October 2021

The following individuals read and discussed the dissertation submitted by student Golam Dastoger Bashar, and they evaluated the student's presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Gaby G. Dagher Ph.D.	Chair, Supervisory Committee
Nader Rafla Ph.D.	Member, Supervisory Committee
Jidong Xiao Ph.D.	Member, Supervisory Committee

The final reading approval of the dissertation was granted by Gaby G. Dagher Ph.D., Chair of the Supervisory Committee. The dissertation was approved by the Graduate College.

DEDICATION

Dedicated to my family

ACKNOWLEDGMENT

I would like to express my thanks and gratitude to my advisor Dr. Gaby Dagher, for his continuous supervision, motivation, and extensive advice throughout the research. His constant guidance, attention, and time is the reason I was able to proceed this far and complete this research. In Dr. Gaby, I have discovered a person with the highest level of professionalism and a kind heart. Along with his research, Dr. Gaby has also inspired me with his towering personality. I am fortunate to be his student.

I am also incredibly thankful to my committee members, Dr. Nader Rafla and Dr. Jidong Xiao, for their outstanding support and precious advice during this journey.

Last but not least, I would like to express my deepest gratitude to my parents. Heartfelt thanks to my wife, Sabiha Alam, for being cooperative and for her constant love, support, and motivation during my PhD journey. Finally, thanks to my sweet daughter, Aliza for accelerating my happiness.

ABSTRACT

In blockchain technology, consensus protocols serve as mechanisms to reach agreements among a distributed network of nodes. In this work, we propose three novel protocols for permissioned, healthcare, and supply chain blockchain.

(1) Proof of Queue (PoQ), for private blockchains, combines the lottery strategy of PoET with a specialized round-robin algorithm where each node has an equal chance to become a leader with equal access. PoQ is relatively scalable without any collision. Like PoET, PoQ uses Intel SGX, a Trusted Execution Environment, to generate a secure random waiting time to choose a leader and fairly distribute the leadership role to everyone on the network. Our analysis and experiments show that PoQ provides significant performance improvements over PoET, and its fairness scales linearly with the number of SGX nodes in the network.

(2) ACCORD, a quorum-based multi-leader protocol for health record management that achieves fork-resistance, robustness, and scalability. ACCORD consists of three distinct components: (a) an asynchronous quorum selection procedure to designate the creators of future blocks, (b) a block creation protocol run by the quorum to prevent omissions in the presence of honest quorum members, and (c) a decentralized arbitration protocol to ensure consensus by voting. We define the threat model and perform security analysis on the protocol. We also implemented the protocol and conducted experiments to demonstrate effectiveness of the protocol.

(3) In response to the Drug Supply Chain Security Act (DSCSA), we introduce Janus, a novel pharmaceutical track-and-trace system that utilizes blockchain and cloning-resistant hologram tags to prevent counterfeits from entering the pharmaceutical supply chain. We designed a multi-quorum consensus protocol that achieves load balancing across the network. We perform a security analysis to show robustness against various threats and attacks. We implemented Janus, and the experimental results show that the system is fair, scalable, and resilient.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

As a society, we have become too dependent on computer technology. Computers and other devices were introduced to be relied on, carry out tasks faster, act as projects, and be available when requested. However, the set of devices that form the computer network cannot be fully trusted, whether the components are local or global. The distributed consensus mechanisms enable a group of entities to agree on a specific subject despite the failures of a limited number of components and asynchrony. When consensus is achieved, we can build reliability in a network that has multiple unreliable components. Online trade is based on trust among those who are involved in the network. The participating parties often do not know whom they interact with; thus, traditional online selling or exchange is a substantial risk. Here a third party is trusted by the involved parties in a way that all of them will maintain details of the transaction. By doing so, the system achieves authenticity, non-repudiation, and integrity. However, trusted third parties are costly. Blockchain transfers this trust from a third party to a decentralized system consisting of many nodes. In summary, a blockchain is a database that keeps a record of all transactions that occurred within the network and is replicated at each involving node. The principle of a blockchain is to allow trusted computation among a set of mutually distrustful nodes. The idea of blockchain was first presented by the famous bitcoin white paper written by Satoshi

[69], whose true identity remains unknown to date. Due to replications of the transaction to all participating nodes, blockchain maintains data integrity. The paramount part of a blockchain is the consensus protocol, which confirms a common consent on the ledger's state and resiliency of the ledger. The goal of consensus protocols is to authorize the globally accepted group of transactions. Simply, consensus protocol ensures the steady operation of a blockchain system. In blockchain systems, consensus protocols are the cornerstone of achieving security and scalability. They not only enable nodes in the network to agree on the valid information that can be added to the ledger while keeping all nodes synchronized, but they also establish the sequence of order in which blocks (and consequently transactions) are affixed.

The most popular consensus protocol is proof of work (PoW) [69], which is broadly chosen by cryptocurrencies like BTC (bitcoin) and ETH (Ethereum), and suffers from large consensus delays and requires a large amount of computing power. As the BTC network keeps growing, PoW has resulted in different performance bottlenecks and sustainability problems. For example, BTC adopts PoW with a maximum transaction capacity of 3-7 transactions per second (TPS), which greatly limits the application prospect of PoW in actual payment. In contrast, the VISA has approximately 65,000 TPS with 50 million participants. However, PoW is known as an energy-hungry protocol that consumes a massive amount of electricity. Due to these shortcomings, researcher have been proposing many alternative consensus protocols such as Proof of Stake (PoS) [56], Proof of Burn (PoB) [12], Proof of Activity (PoAc) [17], Proof of Capacity (PoC) [39], Proof of Importance (PoI) [10], Proof of Authority (PoA) [33], Proof of Elapsed Time (PoET) [28] [25], Proof of Luck (PoL) [66], etc. for the public, private and consortium types of blockchain network. PoS (suffer from 'bribe attack',

'nothing at stake' problem and makes rich richer), PoB (wastes resources needlessly), and PoET (collision problem) do not demand computation-intensive mining, thus effectively reducing energy consumption. However, they are still way behind in terms of TPS, fault tolerance, and security features. On the other hand, PoC (wasting space) and PoAc requires a large amount of computation power which is a misuse of resources. As a result, a number of researchers have been working to develop a fair, scalable, and efficient consensus protocol for blockchain.

1.1 Research Directions

1.1.1 Research Direction I : PoQ: A Consensus Protocol for Private Blockchains Using Intel SGX

Motivation. In leader-based consensus protocol, a node in the network has some special power to propose the next block in the distributed network. Leader election is an effective mechanism for enhancing efficiency, minimizing collaboration, oversimplified design architecture, and minimizing operations. A single leader-based system can work more efficiently because they are only responsible for handling the next block rather than establishing consensus. Undoubtedly, a single leader-based system can minimize the communication cost and give better performance, which is desired in a private blockchain.

Challenges and Concerns. Several traditional leader-based consensus protocols exist in the sector of blockchain. Some protocols may be scalable to vast numbers of nodes but suffer from poor throughput. However, none of the current protocols can obtain the following characteristics altogether: higher throughput, scalability, fairness, equitably distributed mining, and energy efficiency. On the other hand,

leader-based consensus can raise issues like failure modes or trust that can lead to a more complex scenarios to evaluate the correctness of a system. However, a well-designed leader-based protocol can handle the *stale block* issue effectively. A stale block is an accurate, previously announced block that is not part of the longest chain. A stale block appears at any time when more than one node announces a valid block within a short duration. Some of the existing leader-based protocols are not free from the collision either. A collision occurs when more than one node is eligible to become a leader and attempts to create blocks simultaneously. Existing protocols like Proof of Luck and Proof of Elapsed Time use special hardware called software guard extension (SGX) to achieve minimal power consumption and fairly distributed mining. To have these properties, a protocol is also needed to be designed using SGX.

RQ1. Can a trusted execution environment be utilized to design a leader-based permissioned consensus protocol with high throughput, low energy consumption, and equitably distributed mining?

1.1.2 Research Direction II: ACCORD: A Scalable Quorum-based Consensus Protocol for Healthcare Blockchain

Motivation. Due to the nature of blockchain, it is feasible to implement it in a sector other than cryptocurrency. Healthcare can be considered as a suitable example for such a sector. The motivation for conducting this research is to analyze a model based on blockchain technology and to share and store health data in a more secure way by maintaining full privacy and giving the owner full control of his data. Patients and hospitals must have a reliable mechanism for verifying and validating the actual

health data. A study in 2014 discovered that 15% of patients who visited a care provider in the US documented that they had to bring test results to an appointment individually, and 5% were required to have a repeated test due to the unavailability of previous results [74].

Challenges and Concerns. Some of the main characteristics of a consensus protocol that needs to fit a medical blockchain are energy efficient, throughput consistent, scalable, fork resistance, and fairness. According to [4], health records contain sensitive data and are considered 10x more valuable to hackers than credit card information, thus, the records should not be accessible to the public. Therefore, healthcare consensus is unique from the case of cryptocurrency, which is public so that there is no authentication scheme for participants. There are only two consensus protocols that exist (to the very best of our knowledge) specifically for the healthcare domain. Due to the characteristics of a healthcare blockchain, a quorum-based consensus protocol is a good-fit rather than a leader-based. Also, architecture for a quorum based system has high complexity compared to a single leader as it concerns about other's node's current state.

RQ2. Can we design a consensus protocol for a public-permissioned healthcare blockchain that is fork-resistant, fair, and scalable?

1.1.3 Research Direction III: JANUS: Toward Preventing Counterfeits in Pharmaceutical Supply Chains Utilizing a Multi-Quorum Blockchain

Motivation. The consecutive issue of counterfeit drugs in the pharmaceutical industry has exposed the essence of blockchain in the supply chain. These illegitimate

products cause harm to end users and wreak havoc on the supply chain itself, costing billions of dollars in profit loss. Also, the traditional drug supply ‘ chain management suffers from several other issues such as no end-to-end visibility, delay, fraud, etc. More specifically, the current pharmaceutical supply chain (PSC) suffers from a lack of traceability, security, and transparency. These faults ultimately contribute to the presence of illegitimate products in the market. The World Health Organization (WHO) estimates that the presence of counterfeit products in the pharmaceutical market can range anywhere from less than 1% in developed countries to over 10% in some developing countries [73].

Challenges and Concerns. Starting November 2023, the Food and Drug Administration (FDA) will demand stakeholders in the pharmaceutical supply chain to comply with an act of guidelines called the Drug Supply Chain Security Act (DSCSA) [44]. The purpose of the DSCSA is to generate a computerized track-and-trace system for drugs in the PSC. Holding an electronic system to track particular drugs throughout the PSC can significantly decrease the number of counterfeits in the market. While blockchain can be used to form this immutable electronic system, the challenge of verification of physical products with digital data arises. It also poses the challenge of ensuring end-to-end visibility.

RQ3. Can we design a consensus protocol for a pharmaceutical supply chain system that prevents counterfeits from entering the system as well ensures integrity of delivered products between stakeholders in the supply chain?

Table 1.1 presents an overview of targeted security properties across all research directions.

Table 1.1: Targeted security properties accros all research directions. Symbols: ✓means yes, and ⊘ means not specified.

	Fairness	Fork resistance	Collision resistance	Fault tolerance
RD I: PoQ	✓	✓	✓	⊘
RD II: ACCORD	✓	✓	⊘	✓
RD III: JANUS	✓	✓	⊘	⊘

1.2 Organization

The dissertation is organized as follows:

- Chapter 2 is an in-depth literature review of the existing consensus protocols in cryptocurrencies, healthcare, and supply-chain sector.
- Chapter 3 introduces the preliminaries required for the protocols to establish the background knowledge needed for this dissertation work.
- Chapter 4 provides details of leader based consensus protocol for private blockchain by utilizing SGX.
- Chapter 5 provides the proposed approach to address the problem discussed in research direction II.
- Chapter 6 describes our proposed solution for supply chain management.
- Chapter 7 concludes our work followed by a discussion about future work.

CHAPTER 2:

LITERATURE REVIEW

Distributed computing systems often employ consensus protocols to help reach agreements and make decisions. The concept of reaching consensus was originally motivated by Leslie Lamport's work in 1978. While he was working on event ordering in a distributed computing system [58], he proposed a protocol that required all network participants to ensure that decision-making consensus was reached. The next landmark publication was in 1982 by Lamport *et al.* [60] who laid out the *Byzantine Generals Problem*, in which a quorum of Byzantine generals surrounding an enemy city depended on instant communications and a majority rule to execute military plans. The problem is how each general agrees on the same decision whether to attack or not, where some of them might be malicious, faulty, or disloyal. Each general has the right to strike but an indifferent strike would be baneful. While Lamport's work was able to handle certain levels of malicious actors, it provided no solutions to asynchronous communications. This shortcoming (termed the "Byzantine Failure"), was explored in 1985 by Fischer *et al.* [42] in which the generals were able to communicate asynchronously. It mathematically described instances of delayed messengers and proved that deterministic protocols could not reach consensus if the communications were asynchronous. Castro and Liskov [24] expanded on this work by describing how Byzantine Fault-Tolerant (BFT) protocols could be implemented

when nodes produce arbitrary data. BFT can manage approximately 33% of nodes being adversarial. Martin and Alvisi [64] further whittled this scenario down in 2006, describing cases in which consensus could be reached asynchronously in just two steps. In 2008, through his Bitcoin whitepaper [69], Satoshi introduced blockchain and proposed Proof of Work (PoW), the first fault-tolerant consensus protocol for blockchain, and offered financial incentives to promote node honesty. Blockchain technology cuts out the need for a third-party in many cases. It can be characterized as a peer-to-peer (P2P) network in which participants maintain a ledger of previous transactions, and cryptographically reference their ledger in successive transactions.

Depending on the use case, a blockchain can be either *permissionless* (public), or *permissioned* (private or consortium). In a public blockchain, anyone can join the network, send transactions, create and validate blocks. On the other hand, a permissioned blockchain is operated by known nodes, who are legitimized to validate a transaction. It can be treated as a closed system. Thus, it allows greater privacy than a public blockchain, where the identity of users is detectable but the transactions remain private. Hyperledger is a prominent example of a permissioned blockchain. A consortium blockchain is partially private, functioning under the guidance of a group, where access to the chain is determined by a group of pre-selected nodes, not an individual.

Scope: In this work, we seek to explore prominent consensus protocols in the top cryptocurrencies, discussing their use cases, as well as their relative weaknesses and strengths. Figure 2.1 provides a summary of consensus protocols in the top 50 cryptocurrencies.

We draw from Conti *et al.* for their work in identifying and assessing threats

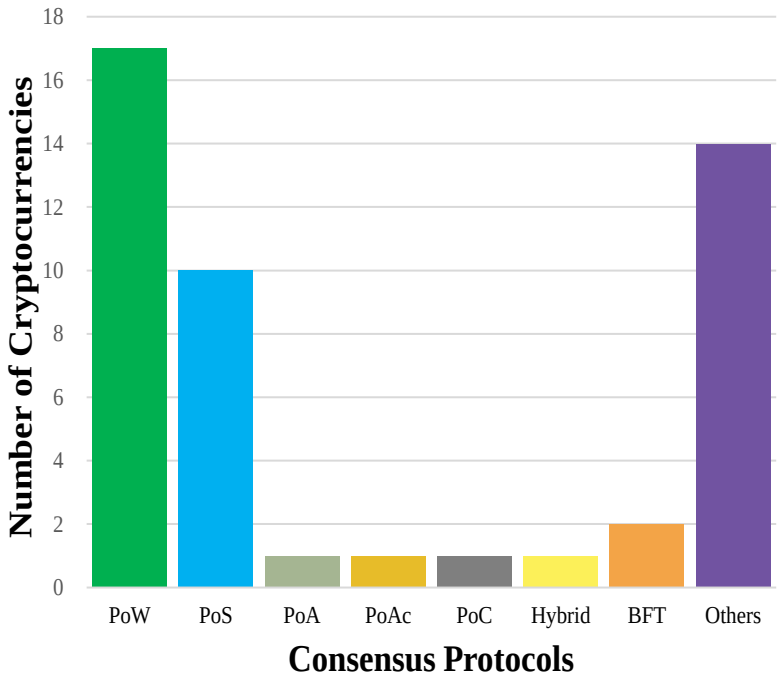


Figure 2.1: Consensus protocols in top 50 cryptocurrencies by market capitalization from CoinMarketCap (April, 2019).

to the Bitcoin network [27]. We also rely on works from Baliga *et al.*, who carried out a comparative discussion of protocols' security attributes and functionalities [11]. Besides, there exists three related works. In [21], the author provides an introduction to consensus protocols while [20] focuses on quantitative analyses for some of them and derive three key components of Bitcoin's design. Last but not least, [13] provides an overall picture of the most important consensus protocols and their working principle. Our work covers large spectrum of existing consensus protocols and attempts a comparative evaluation of their attributes. We believe that identifying a broader set of protocols will allow for a deep comparative understanding of how blockchain technology is being implemented today.

2.1 Taxonomy of Consensus Protocols

We examine the regulatory mechanisms that allow consensus protocols to function. In this work, we focus on cryptocurrency-related consensus protocols. Similar to Bitcoin, Ethereum currently relies on PoW to achieve consensus about its transactions. However, it plans to switch to a PoS algorithm called *Casper* which is still in development. In Ethereum, miners who effectively solve the puzzle receive Ether as a reward. EthHash is a PoW-based protocol performed by the Ethereum network. It uses a different protocol than Bitcoin: the Greedy Heaviest Observed Sub-Tree (GHOST) consensus protocol, a modified version of the Nakamoto consensus, where the heaviest sub-tree wins rather than the longest chain. In the Casper design, validator nodes are required to lock up a certain amount of tokens as a security deposit, and they are then able to cast votes on upcoming blocks that are weighted by their stake. This protocol avoids arbitrarily wasting electricity (as in PoW) and is more decentralized. In the Casper algorithm, an attacker can be identified and their deposit can be overthrown instantly. Besides that, the PoS falls mainly into two categories: Chain-based and Byzantine Fault Tolerant (BFT). In chain-based, validators are chosen pseudo-randomly, and given the privilege to validate the next block. In Byzantine-style PoS, validators are randomly assigned the right to suggest the next block and every validator casts their vote for a round and then multi-round voting decides which block is ultimately affixed to the chain. Because consensus does not rely on the length of the previous chain, it can be accomplished by validating a block in each round. The taxonomy of cryptocurrency consensus protocols is shown in Figure 2.2.

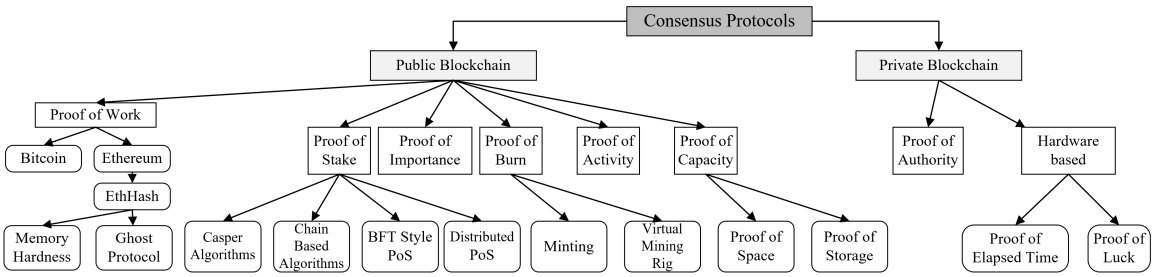


Figure 2.2: Consensus protocols categorized based on blockchain's type.

2.2 Cryptocurrency Consensus Protocols

2.2.1 Proof of Work

The Proof of Work (PoW) concept existed before Bitcoin. It draws inspiration from a 1993 work by two academics, Dwork and Naor, who described a protocol to hinder spam by forcing email users to compute functions before sending an email [38]. However, the term "Proof of Work" was first introduced in 1999 by Jakobsson and Juels [52]. According to the Bitcoin white paper [70], PoW is a combination of computational power and cryptography. It starts by picking out a mathematical puzzle (also called challenge string (CS)), which miners repeatedly hash with a nonce until the resulting hash starts with a certain number of zeros. When the right nonce is found, the miner announces it to the whole network, where nodes can easily check its correctness based on the values in the block. If it is correct, that particular node receives a block reward, and the new block is appended to the public blockchain.

The difficulty of the puzzle increases proportionally to the amount of computing power in the network. That is, more miners in the network makes it more difficult to mine the next block. The hashing sequence is arbitrarily difficult and is known as the difficulty rate. This hashing exercise is thought to ensure network security because the burden of computational work represents an insurmountable challenge to

malicious actors.

The Computational difficulty is calculated at regular intervals. In Bitcoin, it is set every 2016 blocks. PoW rewards miners that have modern, expensive and powerful equipment (ASIC's or GPU's) since such hardware supports fast computation of hashing. To increase their chances of finding the right hash, miners usually join mining pools to team up and put their hashing power together and distribute the rewards among them. However, the pooling approach could lead to the network becoming more centralized. If some of the larger mining pools join together, they could start accepting invalid transactions. If a group of miners acquires 51% of the overall hashing power, it can practically control the blockchain. This type of attack is referred to as the 51% attack, and would lead to the double-spending problem. That is, the attacker could create multiple transactions with the same coins. Since the attacker is spending more coins than available, these transactions should be deemed invalid. However, if the attacker holds a majority of the hash power, such transactions could be treated as valid transactions.

On the other hand, even if a miner successfully mined a new block, the block award is not guaranteed. Let's consider two honest miners A and B who solve a puzzle at the same time and create a temporary fork in the blockchain network. In this situation, the network needs to carry on with anyone of them. The network goes with the chain that has done most of the work. Another weakness of PoW is the slow confirmation rate. BTC, Ethereum, Litecoin and Bitcoin Cash that follow PoW have 7, 15, 56 and 60 transactions per second (TPS) respectively, compared to VISA with 24,000 TPS and PayPal, 193 TPS , which demotivates their use for e-commerce.

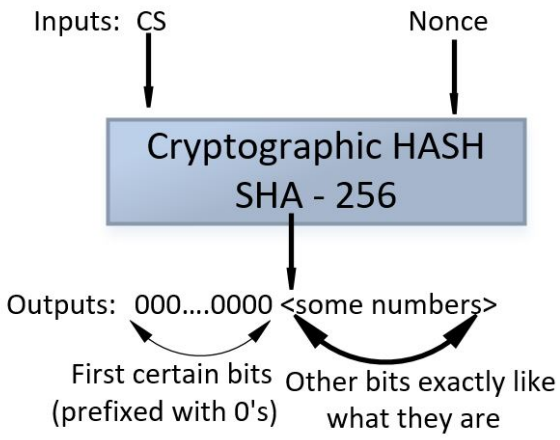


Figure 2.3: PoW Hash Model

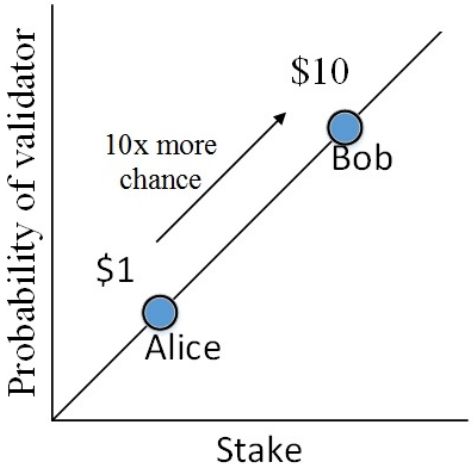


Figure 2.4: PoS Probability

2.2.2 Proof of Stake

Proof of Stake (PoS) was proposed on the bitcointalk forum back in 2011. In PoS, one node is selected in a random way to validate the next block, which depends on its wealth, also called "stake". Here, consensus nodes are known as "validators". To qualify as a validator, a node must deposit a particular amount of coins to the network as stake. The probability of being chosen to forge the next block depends on the size of the vested coins like a linear correlation as shown in Figure 2.4. In PoS, nodes mint or forge new blocks instead of mine. If any validator verifies a fraudulent transaction then they loses a portion of his stake as a penalty which is higher than the transaction fee, thus they gives up more than he earns. In this way, PoS demotivates validator to do dishonest transactions. If any validator wants to leave the network, his stake and the fees from the transaction will not be discharged immediately but after a certain amount of time, because the network still needs to verify all the block transactions that he did.

In PoS, not every validator goes for the new block thus the protocol uses signifi-

cantly less energy. No mining pool concept exists here, hence it is more centralized compared to PoW and also does not require any expensive mining equipment. Peercoin and Nxt implemented by using pure PoS protocols. The TPS of PoS cryptocurrencies are relatively faster compared to PoW currency. BlackCoin and Novacoin implement age-weighted PoS protocols in which staked tokens are more likely to become validators based on the age of their deposit into an address. Under the PoS, each coinage is seized in the term of "coin-day", so occupying 100 coins for 100 days is equal to 10000 coin-days. The value is reset to zero if the coins are utilized. A node is considered as a validator by consuming its coinage. BlackCoin's age-weighting function is based on the given expression:

$$proofhash < coins \times age \times target \tag{2.1}$$

Where, the *target* is the mandatory number of coins set by the network through difficulty adjustment, *age* is the coinage, *coins* is the number of coins a validator has used to be chosen and the *proofhash* is the unused output.

51% attack becomes impractical in PoS because of cryptocurrency's value. In PoW, this attack can be hard to detect since the hash power can be distributed among different miners. However, it can be easily detected in PoS, since the market can recognize that a large portion of coins have been bought and are in control of a single entity. In PoS, a miner can mine both forks because it costs nothing; thus choosing a wrong fork is not costly and at any time they can switch with the winning fork. This can be lead to a problem called nothing at stake. Let's assume party P_1 sends some money to party P_2 and in return, he wants something. P_1 does an honest and dishonest transaction. The honest transaction sends cryptocurrency to P_2 and

dishonest sends cryptocurrency back to P_1. For this, there will be a fork. As miners can mine in both forks, if P_1 is also a miner and does mining upon the dishonest fork, it would still be validated even if it is dishonest. This specific problem has yet to be overcome in Casper algorithms.

2.2.3 Proof of Burn

Proof of Burn (PoB) [13] is an alternative consensus protocol proposed by Ian Stewart that requires a participant to burn a mined cryptocurrency to get privileges. PoB tries to fix the power consumption issue of PoW and relies on its cryptocurrency, which is called "bootstrapping" coins. A participant needs to send cryptocurrency to an unspendable address which serves as the proof that the coins have been burned. By doing this, those coins get out of circulation and can no longer be spent. Therefore, PoB is often called PoW without energy waste. The principle of PoB is that the miners burn the virtual currency tokens, to get the right to write new blocks in proportion to the coin burned. Thus, the inventor cites an analogy "Burnt coins are mining rig". Burned coins are also documented on the blockchain, to ensure that the coins cannot be turned to account again.

Stewart described two blockchain system in which users could irretrievably send tokens from blockchain A to an "eater" address, and receive corresponding tokens in blockchain B. The eater address would have a non-randomly generated a public address and consequently have an unknown and unlikely-to-guess private key. Chain B would then function as a network in which the chain A tokens operated as miners with hashing power proportional to the token quantity and block height of the burn transaction in chain A. The more coins a miner burns, the higher the chance he has of being selected to mine the next block on the new currency's blockchain. Several coins

are using PoB as a consensus protocol. Slimcoin (SLM), Counterparty (XCP) and Triggers (TRIG) are the best-known examples. PoB is expensive for an individual miner, however, it is considered as an investment for the future which leads to greater stability. As there is no guarantee for a miner to recover the value of the coin burned, PoB is a greater risk.

2.2.4 Proof of Activity

Proof of Activity (PoAc), first proposed in 2012 combines the idea of both PoW and PoS. It acts as a middle ground between them, combining the computational cost of PoW mining with the slimmed-down staking method of PoS consensus. The mining part kicks off in the standard PoW process. Once a block is found, then it follows PoS principals, where the new block only contains the header and the address of the winner. Depending on the header details, a group of validators is chosen randomly who are required to sign the new block. The more cryptocurrency a validator owns, the more chances he has for being selected as a signer. After being signed by all the validators, the newly founded block turns into a full-fledged block and gets appended on the blockchain network. It requires the stakeholders to be online at all times to participate, hence the *activity* part of the name which requires continuous exchanging of data. Espers and Decred are only two coins that use PoAc. Decred is considered a fully-implemented example of PoAc consensus. PoAc protocols require high computational overhead during the PoW phase, and whoever has more cryptocurrency still has a higher probability of getting into the signer's list. However, 51% attack goes down dramatically because a successful attack requires the same miner to control both 51% mining hash rate and a majority of the cryptocurrency.

2.2.5 Proof of Capacity

Proof of Capacity (PoC), also known as Proof of Space, requires a competitor to pay something to get in the competition. The idea was devised by Dziembowski et al. in the Proofs of Space whitepaper [39]. The concept of PoC is identical to PoW, except instead of computation, storage is used. It differs from PoW by plotting which is used to verify the block. In general, if a lottery reward based on matching the most numbers on winning tickets, then the participant who fills his HDD with lots of lottery tickets has a higher probability to win the lottery. The more solutions, the higher the chance to win. Plotting uses a hash algorithm called Shabal, which is relatively slow. Since it's difficult to compute, the miner needs to compute and reserve the result on the HDD before the competition begins. This procedure is called plotting the hard drive. The next step is mining, consisting of miners reaching the solution, whoever solves it first needs to mine the next block. Burstcoin already adopts PoC. Since it does not require the involvement of CPUs and GPUs, it is naturally very low on power usage. PoC uses HDD where data can be compromised. Anyone with a regular hard drive can participate in this protocol which is 30 times more energy-efficient than ASIC based mining. Everyone who has a hard drive can mine, even from a normal android phone's making the network more centralized. Here, no updated model of HDD is required because an old HDD can store the same data as well as new ones.

2.2.6 Proof of Importance

Proof of Importance (PoI) was launched by the New Economy Movement (NEM) in 2015. In the PoS, participants interested in saving instead of spending. But, PoI rewards not only the largest amount holder but also how much they transact and with whom they transact. That means PoI gives the priority to the person who directly

helps the economy and continuity of the blockchain. PoI demotivates the concept of "rich gets richer". Everyone in the network gets the chance to be rewarded based on how much effort they put forth. Here, the method of adding new blocks to the chain is called harvesting. An important score is assigned to each account on the NEM network. This score will determine one's ability to harvest within the network. To be part of the score calculation, the user account needs a minimum of 10,000 XEM (cryptocurrency related to NEM) in his account. The current supply in the circulation of XEM is 9 billion so it is not expensive to participate. As participation grows, the 10,000 XEM threshold will be changed to accommodate the increase in activity. After qualifying, a mathematical calculation will be used to determine the current score in the system. Two important factors used in scoring are Net transfer (transaction happen in the last 30 days, in-fact recent transaction weighted more heavily) and currency that is vested for creating blocks. So the more transactions associated with one's account over a period, the higher the importance score. It's not like an individual has some addresses and passes XEM between them to increase his score. The way it handles this is by lowering the importance score for those accounts who send out and then receive XEM. Purchasing goods via XEM won't increase the importance score too. XEM has faster TPS than others. Since PoI is similar to PoS, it does not require heavy computation and cost is relatively low since the only expense is score calculation. Nothing at stake attack is possible because the creation of a block costs no resources, wherever there is a fork, someone can freely create blocks on both forks and 51% attack is possible. But when performing the attack in PoI, the attacker needs to take into consideration the transaction activity involved with their account in addition to the number of coins owned.

2.2.7 Proof of Authority

Gavin Wood, the co-founder of Ethereum, first termed Proof of Authority (PoA) which is a permissioned based consensus protocol and considered as a modified version of PoS. Here, one's identity is staked while in PoS, one's coins are staked. The true identity of a person is validated through notaries where identity information shares publicly and go through an on-chain verification using smart contracts. Due to a background check it's difficult to have a license in the notary. A participant needs to invest his money as well as reputation to be a validator. PoA depends on a finite number of validators that ensures more scalability and security. PoA is basically designed for Ethereum which has faster block generation. It includes Hyperledger and Ripple. Hyperledger follows PBFT. On the other hand, Ripple is using iterative processes. PoA is super-centralized but efficient, compatible with private blockchain. Since PoA is automated, heavy computation and regularly monitoring is not required. Even a Raspberry Pi can be used to achieve most of this objective. Compared to other protocols, the cost is relatively low since the required equipment is not expensive and also does not require to communicate among the nodes to reach a consensus. PoA is less vulnerable since it requires the validator's account to be authentic. But when staking, their real identity is at risk which acts as a barrier to acting maliciously. PoA consensus protocol is utilized in Ethereum's Kovan testnet and VeChainThor blockchain network. Microsoft Azure is another example of where the PoA is being implemented. PoA is a fast and high TPS.

2.2.8 Proof of Elapsed Time

Proof of Elapsed Time (PoET) is a consensus protocol that seeks to curb the wasteful use of resources exhibited in PoW. Instead of wasting lots of energy, it

achieves consensus with a fair lottery system [25] to elect a validator node. It was developed at Intel in early 2016 as part of their work on the Hyperledger project. The PoET protocol is intended for a permissioned blockchain, in which participation is access-controlled, and in which node identity is disclosed before participation. The backbone of the PoET protocol is Intel's proprietary hardware, ***Intel SGX***, which houses cryptographic primitives such as random number generator SGXRND. Depending on the pillar of a fair lottery system, each node is equally likely to be a winner. In PoET implementations, `sgx_read_rand()` functionality assigns a waiting period to each participant and the node with the lowest waiting time becomes the validator.

In general, each node will obtain a timer value from the trusted code and wait for the obtained timer value. The node to finish first, add a new block to the blockchain and broadcast the essential data to the network. The same procedure then replicates for the finding of the next block. To join a PoET network, the participants must initialize their new key pair and send a join request to the network. Once the participant has joined the network, they can opt to be involved in the lottery process. Random times are allocated to each node, therefore all blockchain functionality is carried out in a determinant process. The SGX hardware is also not run in a way that promotes hashing competition, but it acts as the barrier to entry. PoET also measures every participant's lottery winning occurrences to determine malicious or suspicious nodes and blacklist them. It also needs to confirm two things primarily that each node gets the random time from TEE and it actually waits for that specified period. However, the main concern of this protocol is it depends on SGX which is assembled by Intel, so the dependency of PoET goes to Intel, a third party company.

Table 2.1: Comparative evaluation of attributes among cryptocurrency consensus protocols.

Proof of ...	Type	Difficulty	Barrier to Entry	Platform	Anonymous Mining	Known Issues & Attacks
Work	Public	Difficult	Mining hardware	Bitcoin, Zcash, Ethereum, Litecoin, Bitcoin SV	Yes	Selfish mining, DoS, Sybil [95]
Stake	Public	Easy	Staked tokens	Ethereum, Tezos, Peercoin, EOS, BlackCoin, Gridcoin, Novacoin	Yes	Nothing-at-stake, Bribe, Pre-computing, Short & long range attacks, DoS, Sybil
Burn	Public	Easy	Irretrievable coins	Slimcoin, Triggers, Counter Party	Yes	None
Activity	Public	Difficult	Staked tokens, Mining hardware	Decred, Esper	Yes	Nothing-at-stack (very low chance)
Capacity	Public	Difficult	Data storage	Burstcoin, Permacoin, BitTorrent	Yes	Malicious hard drive
Elapsed Time	Private	Easy	Intel SGX hardware	Experimental stage	No	Betrayal of trust
Authority	Private	Easy	Reputation	Vechain	No	Identity disclosed
Importance	Public	Easy	Money Reputation	NEM	Yes	51%, Nothing-at-stake
Luck	Private	Easy	Intel SGX hardware	None	No	Sybil

2.2.9 Proof of Luck

Proof of Luck (PoL), was introduced fairly recently to overcome the cost and wastage of resources in mining Bitcoin and Ethereum using PoW. Current Intel CPUs have a new set of instructions called SGX. It gives special permissions that allow us to run code inside an environment that is secure and the execution cannot be modified even by the operating system, BIOS or VMM. PoL is similar to PoET because it also generates random numbers referred to as *luck*. Luck is selected by the participants and participant with the highest number gets the winning block and termed is *luckiest*. Since PoL requires secure computations that are dependent on the special instruction set provided by Intel SGX, it acts as the barrier to entry. This also ensures that the luckiest block was selected according to protocol [66]. In PoL, it is difficult to make attacks. For example, attackers need to be lucky as well to make double-spending.

2.3 Comparative Evaluation of Consensus Protocols

2.3.1 Cryptocurrency platform

From the above description of various consensus protocols, we compare all the protocols in terms of their network type, computation, power consumption, expense, wastage of resources, platform (cryptocurrency), anonymous client and validators, participation and known issues. We refer to some indexes to show its properties. *Public* blockchain means, it is accessible by all, and on the other hand, *private* means it has some restrictions. Computation is expressed by "difficult" or "easy", which means whether the protocol requires heavy computation or not. For anonymous mining, if a protocol requires the identity to be disclosed then it is indexed as "no" otherwise "yes". Finally, based on various criteria, well-established protocols are vulnerable to some

attacks, such as 51% attack, DoS attack, Sybil attack, Short and Long-range attack, Pre-computing attack, and Nothing-at-stake attack. A comparative evaluation of cryptocurrency-based consensus protocols is shown in Table 2.1.

2.3.2 TEE platform

In recent years separate approaches are used to extend the performance of *PoET*. Research has been conducted to achieve a good overall performance in a private environment. In addition, there are existing related works that spotlight the perfections behind the intention of Trusted Execution Environment (TEE) design. Hardware-based TEE like ARM TrustZone (available on smartphones) or Trusted Platform Module and Intel SGX (for x86-based computer) are generally obtainable in commodity computing platforms. In the paper [25], authors provide remarks on the design of Sawtooth. In order to reduce potential collision, they discussed waiting times need to be longer. While [32] focusing on reducing the *stale block rate* by restricting the number of nodes which they call *PoET+*. A stale block is an accurate, previously announced block that does not belong as part of the longest chain. A stale block appears at any time when more than one node announces a valid block within a short duration. Proof of Luck (*PoL*) [66] is another consensus protocol based on TEE and similar to *PoET* because it also generates random numbers from SGX into the block referred to as '*luck*' of the block. The protocol selects the chain with the highest accumulative luck as the winner and is determined the luckiest. The luckiest block is then added to the chain. It will generate forks when the network is periodically portioned because the partitions will ensure various largest accumulative luck. In [7], the authors proposed Proof of TEE-Stake (*PoTS*), that leveraging functionality from TEE for public blockchain, where each node in PoTS ensures the same structure to

bootstrap a TEE program. In [31], the authors explore the response of throughput of *PoET* and propose a simple adjustment to it (they termed as "$S - PoET$") which leads to a higher throughput as the network becomes larger. According to the authors, if the shortest `waitTime` and another `waitTime` are conflicted by fewer than the propagation delay then that will result in a stale block.

At the beginning, *PoET* was preferred to replace the *PoW* with the exception of the longest-chain rule [92]. Due to its access control nature, it is most appropriate for permissioned blockchains, where certain works will be executed on TEE by certain authenticated nodes. Unlike permissionless, most permissioned blockchains don't require any rewards mechanism. Each consensus protocol is unique based on the way it creates a block, discloses the evidence (block propagation), the procedure of validation inside the network, and the rewards system for an honest effort. Table 2.2 shows an assessment among some of the protocols designed for TEE based or not. The throughput measurements derive from the complimentary white paper or formal demonstration of the implementation of that protocol and indicate the scales of fastness, i.e. high or low.

Due to dynamic *QT* in PoQ, all nodes are approximately given the same priority to execute depending on its tier (A relationship engaging level between a set of nodes. PoQ is a multi-tier approach to the early recognition of nodes and provides necessary data to them), thus no nodes are left behind which leads to more speed in the system. PoQ progresses in a round-robin way, wherein each round, a selected node within all the nodes in the network will get a chance to reduce its `waitTime` and if it is successful to reduce all its `waitTime`, then that particular node proposes the potential block (a successive set of transactions). Thus, a node cannot get a total allocation of time

above its assigned time. We can say that *PoET* is suitable for a limited network with small `waitTime` while in PoQ arrival time of a node puts great importance on becoming the leader quickly as it maintains a dynamic queue that pushes the nodes based on their arrival time. That means when a node appears in the network it starts to compete with other nodes and the leadership cannot be predetermined. It is suitable for a large number of participants, easy to implement, and also offers similar average `waitTime` and average elapsed time for all its nodes. Due to no hashing is required in PoQ, we can say it also saves energy too. As PoQ is suitable for private blockchain network, it does not provide incentives for participants. To the very best of our knowledge, no work has been done to make the lottery election system fairer for *PoET*.

Table 2.2: Assessment of blockchain consensus protocols. Symbols for binary values: ✓ means yes, ✗ means no. Symbols for non-binary values: ● means high, ◗ means medium, ○ means low and ⊘ indicates undefined in the protocol white paper.

Concensus Protocols	Type		Block propagation	Block Validation	TEE based	Resource consuming	Rewards	Nodes execution	Throughput (TPS)	Fairness
	Public	Private								
Nakamoto (BTC, Litecoin)	✓	✗	PoW	PoW (longest chain)	✗	●	✓	●	○	●
Nakamoto (Etheruem)	✓	✗	PoW (Ethash)	PoW (GHOST)	✗	●	✓	●	○	●
PoET (Hyperledger)	✗	✓	PoET within TEE	TEE certificate	✓	○	⊘	◗	●	●
PoTS	✓	✗	PoTS-based committee election	TEE, eligibility	✓	○	⊘	⊘	⊘	⊘
Chain-based PoS (Nxt)	✓	✗	PoS	PoS (longest chain)	✗	◗	✓	⊘	○	○
PoI (XEM)	✓	✗	PoI Harvesting	Importance score	✗	○	✓	◗	●	●
PoL	✗	✓	PoL	Longest total value of luck	✓	○	⊘	◗	⊘	●
PoQ	✗	✓	PoQ within TEE	Completion of wait time	✓	○	✗	●	●	●

2.3.3 Healthcare platform

Permissioned blockchains are systems where only an authorized group of miners are allowed to participate in the system. This contrasts with permissionless

blockchains, where anyone may participate without any certification or permission. The most familiar instance of this category is Hyperledger and R3 Corda. The Hyperledger Project is a collection of independent blockchain frameworks and tools that offer a wide variety of capabilities. Hyperledger Fabric is a blockchain system, initially designed by IBM, has multiple variants to support a wide variety of implementation requirements [49]. Other examples of permissioned consensus protocols are Proof of Elapsed Time (PoET) [25] and Raft [72].

A permissioned blockchain is, by its very nature, more centralized than a permissionless blockchain [90][15]. This centralization allows the nodes in the consensus protocol to have a comprehensive list of miners at all times. This ability allows for more efficient and structured consensus protocols than permissionless systems can currently provide [90]. However, for many purposes that call for blockchain, this centralization is unacceptable due to the potential corruption of the central authorities. In healthcare, with nominally reputable hospitals managing nodes, this kind of centralization is generally considered to be acceptable [40].

A common approach that becomes available with a permissioned blockchain is the concept of selecting a leader to create a block to reduce redundant block creation work and reduce the number of competing blocks. There are many examples of leader-based consensus protocols [24, 40, 59, 68]. The mechanisms behind these protocols vary widely, but they mostly include three major components: leader selection, transaction acquisition, and block creation and distribution.

The general premise is as follows: the client forwards transactions to a leader chosen from the mining pool, then the leader sends out a block with those transactions. This can result in a more energy-efficient system, as only the leader needs

to construct the block. However, leader-based consensus protocols may encounter problems, as malicious leaders may manipulate the contents of their blocks to their advantage. Many leader-based protocols have mitigations or solutions to this problem [24, 40], but they are often costly, requiring network-wide synchronizations [40]. Our design, ACCORD differs from these approaches since we do not select a single leader. Instead, we divide the role of leader evenly between a group of nodes to improve robustness and efficiency.

Healthcare is a field that is different from cryptocurrency, as cryptocurrency demands a highly decentralized scheme and requires that anyone be able to participate without having to get permission. Healthcare data is different, as it is naturally centralized and requires a certain amount of trust that it was created correctly (e.g., we assume that hospitals run their tests honestly and output the correct results). Hospitals also wish to know whom they are treating. Therefore, a certain amount of authentication and centralization is appropriate. Similar to [36][40][63][85][93], our design requires an authentication scheme to manage identity on the network.

Blockchain technology can be used to support many sectors of the healthcare system. A major use case of blockchain in healthcare is managing electronic health records (EHR), a.k.a, personal health records (PHR). There are many examples of work focusing on EHRs in blockchain [9, 30, 36, 40, 51, 53, 96]. There has been a significant amount of research focusing on secure electronic creation, storage, and management of EHR. Azaria et al. introduced MedRec [9], an Ethereum-based EMR management system in which data permission and operations are recorded in the blockchain. MedRec authenticates participants, store hashes for data integrity, and use smart contracts to interface with providers to view data. MedRec aims to address

issues like response time in data access, interoperability, and better data quality for healthcare research. On the other hand, Dubovitskaya et al. [36] introduced a permissioned blockchain network that uses a cloud-based EMR sharing system for cancer patients. Both of these works have been prototyped but have not been implemented on a large scale. In Ivan's work [51], the author outlined a public blockchain, where patient healthcare data is encrypted but stored publicly, to create a blockchain-based EHR system. In Zyskind et al. [96], the authors have described a decentralized personal data management system that ensures users governance their off-chain medical data (transfers ownership of health records to patients).

2.3.4 Supply chain platform

Many researchers have been studying numerous applications of blockchain since Satoshi Nakamoto introduced the first implementation of blockchain in their famous Bitcoin white paper [69]. Due to its immutable nature, the technology has proved promising for different industries, including supply chains. In addition, it can be exploited to benefit systems in a number of ways, such as boosting transparency between stakeholders, building a decentralized system, and creating a traceable ledger.

Researchers have been working on developing ways to use blockchain to improve the pharmaceutical supply chain [6, 37, 16, 46, 67, 83]. The authors in [37] describe how blockchain can be implemented in the traditional pharmaceutical supply chain system to share information securely. Their proposed design used both local and global blockchains for storing transactions between stakeholders in the network. The local blockchains store transactions between stakeholders of the same type, while the global blockchain stores transactions between the different types of stakeholders. In order to establish consensus, transactions are generated and sent to a validation leader

to be checked. If the transaction is accepted as valid, the validation leader proposes a new block for the remaining validators to vote on. The authors of [6] also utilized a validation leader. However, instead of having both global and local blockchains, they proposed creating a blockchain for each individual product. Products are tracked and traced on the blockchain via near-field communication (NFC) tags. Both systems outlined in [37] and [6] came short of achieving true decentralization. Due to the single-leader-based consensus protocol, their system is vulnerable, as a malicious leader can decide the conclusive order of transactions [55]. Additionally, a single leader can act as a single point of truth and thus a single point of failure. While both [6] and [37] came short in their virtual processes, it is important to note that issues in the pharmaceutical blockchain can occur in the physical aspect as well. To date, the blockchain community has not reached a consensus on which method of labeling products should be preferred to connect the physical product with the digital data on the blockchain. Both [16] and [46] utilized quick response (QR) codes as the medium to protect from counterfeit products entering the supply chain. The usage of radio frequency identification (RFID) tags is applied in [67] and [83]. According to [50] and [91], these tags are vulnerable to various attacks, especially cloning and modification attacks. Additionally, RFID tags are vulnerable to privacy attacks as shown by [41]. Depending on the frequency band used, these tags can be read from distances ranging from 1 to 100 meters [91]. RFID tags and QR codes are cost-efficient but utilize simple technology that can be compromised in a matter of seconds. This poses a huge threat to the PSC and can aid malicious parties in introducing counterfeit products into the chain.

Recently, non-pharmaceutical fields have also started to focus on blockchain as

a potential improvement to their supply chain systems. Food and agriculture safety is one sector that is gaining attention in commercial and academic projects. As of now, most of the solutions are centralized and not free from fraud and tampering. Hence, research has begun to propose different blockchain-based traceability schemes in agri-food supply chain systems. Authors of [82] proposed a system utilizing the Interplanetary File Storage System (IPFS) to store transactional data from the agri-food supply chain while storing the hashes of that data in the Ethereum blockchain. Only authorized users are allowed to participate in the network, which implements a reputation-based system in order to establish additional trust between participants. Their architecture suffers from some shortcomings. Currently, the system lacks a means for returning items or providing refunds. Also, the reputation system has no protection in place to prevent fake or biased reviews.

Blockchain can benefit the pharmaceutical industry as it offers three important features: privacy, transparency, and traceability. Therefore, many researchers have already designed various blockchain frameworks to utilize these properties. Authors of [80] proposed the use of blockchain to keep a transparent ledger of activity for the pharmaceutical research and development process. Transparency on the chain would allow investors access to all previous stages of the research process. Similarly, [61] designed a system in which pharmaceutical products were tracked throughout the manufacturing stage. This can aid in the detection and tracking of counterfeit products from pharmaceutical manufacturers.

CHAPTER 3:

BACKGROUND

3.1 SGX

Intel Software Guard Extension (SGX) is fully implemented in the CPU hardware and yields a partial element to execute within an isolated environment, referred to as an enclave [29]. Generally, SGX breaks down an application into two logical segments: enclave and untrusted part (conventional application). Furthermore, an SGX application can handle 5-20 enclaves. The code in the enclave is used to handle the secret data. On the other hand, the remaining portion of the code, along with all its modules, keeps in the untrusted part. Interaction within these two parts happens via the call gate. A function call that enters the enclave from the untrusted portion is called an Enclave CALL (ECALL). A call within the enclave to an untrusted portion is called an Outside Call (OCALL). Figure 3.1 provides a high-level view of ECALL and OCALL communication. By definition, an OCALL is made from within an ECALL because an ECALL needs to enter the trusted portion. Figure 3.2 shows the execution of an SGX application and the way SGX safeguards an enclave from any envious program, including OS, BIOS, drivers, and firmware which pretends to steal application secrets[1].

[1]https://software.intel.com/en-us/articles/intel-software-guard-extensions-tutorial-part-1-foundation

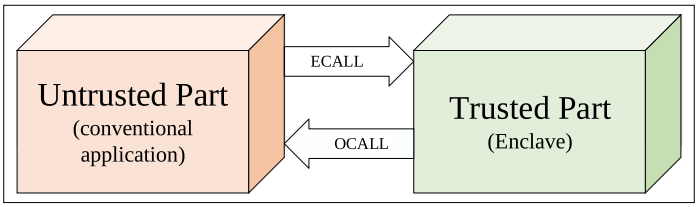


Figure 3.1: Interaction between enclave and untrusted part in an SGX application.

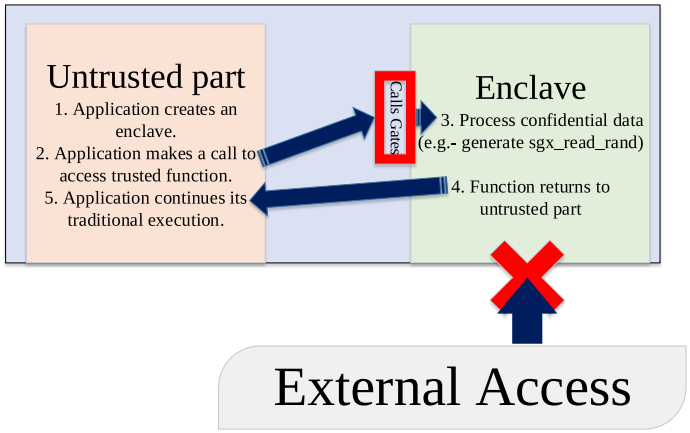


Figure 3.2: Flow of execution in SGX application.

SGX depends on remote attestation to prove to remote users that the particular portion of code is executing in a genuine SGX-enabled CPU [29]. It also presents a reliable source of random number via its `sgx_read_rand` API which calls the hardware-based pseudorandom generator (PRNG) over RDRAND on Intel CPUs. Many researchers have already established that this random number generator is secure and cannot be modified from outside the enclave [26]. Presently, C and C++ are supported by Intel's Software Development Kit (SDK) which is available for both Windows and Linux.

3.2 Abstract model of PoET

PoET is usually used on the permissioned blockchain networks to determine the

Table 3.1: Comparison of SGX based system. Symbols: ● means fully-provided, ✗ means unsupported and ⊘ means unspecified.

	Energy efficient	Higher throughput	Scalability
PoET	●	⊘	✗
PoL [66]	●	⊘	✗
REM [94]	✗	✗	✗
PoQ	●	●	●

leaders of the block on a specific network. After joining the network, each node must ask for a `waitTime` from an enclave and then wait for that randomly chosen `waitTime`. A node who finishes the `waitTime` first – that is, the node with the shortest wait time for an appropriate transaction block, establishes a remote attestation that provides information for verification about its honesty, carries out a new block to the blockchain, and announces the mandatory data to the network. To find the next block, an identical procedure is required. By remote attestation, *PoET* ensures that the nodes select a random `waitTime` (not purposely chosen a curtailed `waitTime`) and the leader has actually waited the allocated `waitTime`.

3.3 Remote Attestation Architecture

Remote attestation (RA) is an exceptional property of Intel SGX, to establish a secure environment between the server and the node (client) [54]. Simply in computing, the term attestation means, a procedure to verify the identity of a software and/or hardware. More specifically, RA is a medium to verify the interaction between the software and the hardware that has been founded on a trustworthy platform. By

following remote attestation flow, a client enclave ensures three things: its identity, its pureness (has not been altered), and a certain piece of code executing in a genuine SGX-enabled CPU. A server sends a remote attestation request to a node and it responds to the request by announcing information about the platform configuration. Node executes the client code while the server runs the server's side code. Both parties are interacting over a network, which is not recognized to be part of any side or secured. The whole operation contains fifteen steps with the server (also called challenger) and the node. Figure 3.3 shows the interactions between the entities engaging in RA. It is worth mentioning that RA adopts a modified version of the sigma protocol to support Diffie-Hellmann key exchange (DHKE) among the node and the server. The sigma protocol is proof that consists of commitment, challenge, and response. SCIFER [5] uses RA to verify the identity of users. Finally, we trust Intel to execute SGX RA service correctly (similar to [5, 94]).

3.4 Holographic Encryption

By utilizing holographic encryption on the physical products in the PSC and linking its data to the blockchain, we can provide a digital traceability scheme for tracking from source to end consumers. This type of tagging system is resistant to various attacks including cloning and modification attacks, making it a secure choice for the pharmaceutical supply chain [62]. Authors of [75] introduced a process for encrypting holographic information utilizing the expanded Diffie-Hellman (EDH) algorithm. By utilizing this structure of holographic encryption on physical products and storing the hashes in the blockchain, JANUS can provide a digital traceability scheme for tracking from source to end consumers.

Remote Attestation Protocol

1. In reply to the challenge request from the server, the node will do the following:
   1a) Initialize the enclave by `sgx_create_enclave (.., enclave_id, ..)` and perform ECALL to go into the encalve.
   1b) Initialize the RA flow by calling `enclave_init_ra(enclave_id,..,b_pse, context)`. Here, *pse* means platform service.
2. If *b_pse* is true then call `sgx_create_pse_session()` before establishing the RA and key session.
3. Call `sgx_ra_init(&sp_pub_key,b_pse,context)` by passing the server's public key. Key is in little-endian byte order and must be hardcoded into enclave.
4. Close PSE session by `sgx_close_pse_session()`.
5. Return *context* to the untrusted part from the enclave.
6. The untrusted part of the node call `sgx_get_extended_epid_group_id()` to get active extended group ID (GID) of enhanced privacy group ID. EPID is an anonymous signature scheme for attestation.
7. This is send to the server as a body of msg0.
   7a) Verify by the server. If it is not valid, server terminate the attestation flow.
8. The untrusted part of the node calls `sgx_ra_get_msg1(...,enclave-id,g_a)` where *g_a* is a public key of a node enclave and this *enclave_id* is going to be attested.
9. The untrusted Key Exchange (uKE) part of the node builds a message, msg1 that contains g_a || GID.
10. Send msg1 to the server. All elements of msg1 are in little-endian byte order.
    10a) Server translate all elements into little-endian order to check.
11. Server replies with msg2 that contains *g_b*, *spid*, *quote-type*, *kdf-id*, *sigRL*, etc. The public key of the server, also known as g_b is based on NIST-256. Signature Revocation List (sigRL), is a list of unfaithful signatures, signed by the revocation authority.
12. After receiving msg2, the untrusted part calls the function `sgx_ra_proc_msg2(context,enclave_id,sgx_ra_proc_msg2_trusted_t, sgx_ra_proc_msg3_trusted_t, msg2, msg2_size, ...)`
    12a) By calling `sgx_ra_proc_msg2`, node builds msg3.
13. `sgx_ra_proc_msg2()` builds msg3 that contains mac, g_a, and platform security property.
14. The node sends msg3 to the server and expect to get the attestation result.
15. Upon receiving msg3 from the node, the server will do the following:
    15a) The server verifies the msg3 by calling `sgx_ra_proc_msg3_req(msg2,msg3_size,att_result_msg)`, to compare g_a w.r.t. g_a of msg1 and verify the msg mac using sigma protocol (SMK).
    15b) Send attestation result message to the node.
    15c) The node will receive the result and checks the MAC using MK.

Protocol 3.1: Remote attestation

3.5 Blockchain Network Types

To ensure a safe and trusted blockchain network, there are varying levels of privacy that can be applied to the network. Commonly-used privacy levels are public,

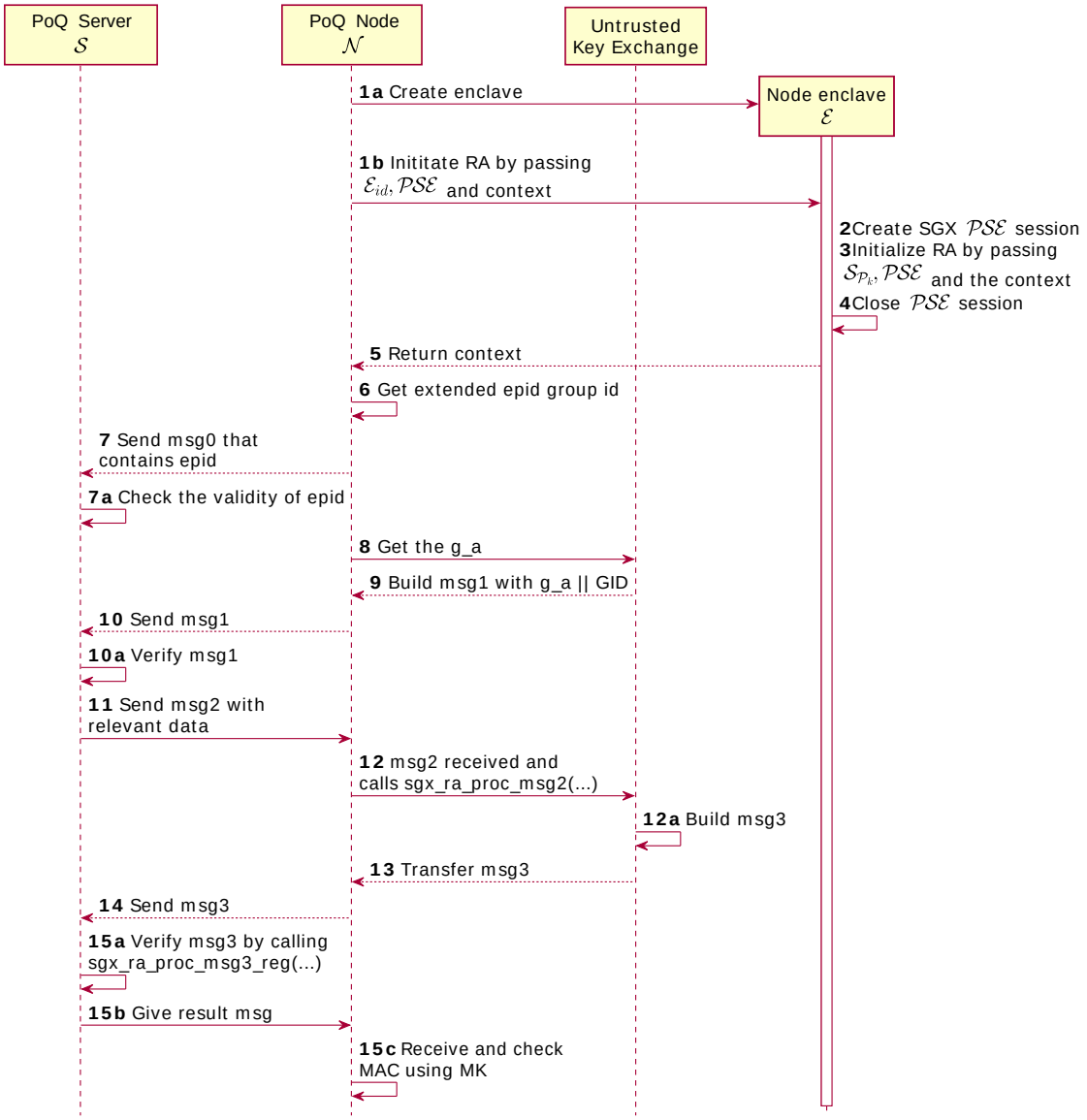


Figure 3.3: SGX Remote Attestation.

public-permissioned, and private. In a public blockchain network, transaction visibility is public and open to anyone. The most well-known implementation of a public blockchain is the Bitcoin ledger [69]. We define public-permissioned blockchains to be where the ledger is available to view by anyone, but participation requires authorization. On the contrary, private blockchains preserve the most amount of privacy as they

cannot be viewed or contributed to without proper credentials. Private blockchains are more applicable for sensitive systems such as health care or banking, where patient and customer data is valuable and confidential [15].

3.6 Drug Supply Chain Security Act (DSCSA)

In 2013, Congress enacted the Drug Quality and Security Act, which introduced the Drug Supply Chain Security Act (DSCSA). Its goal is to negate illegitimate products from the pharmaceutical supply chain [44]. Stakeholders in the chain must follow guidelines that will increase the security of the system. These guidelines include electronic submission of transactions, annual proof of licensure of warehouses and third-party logistics providers, and specific labeling rules [44]. In November 2023, the DSCSA will be in full effect and stakeholders will be required to comply with its rules and regulations. In compliance with the DSCSA, our system includes package-level labeling that allows for a more strict track-and-trace system. Blockchain would be an ideal solution for the pharmaceutical supply chain to seamlessly follow DSCSA guidelines as it creates an immutable ledger of electronically-submitted transactions, further providing a secure track-and-trace system.

CHAPTER 4:

POQ: A CONSENSUS PROTOCOL FOR PRIVATE BLOCKCHAINS USING INTEL SGX

4.1 Introduction

At the core of any blockchain platform, there is a ledger that is maintained by a trustless P2P network. Due to the untrustworthy nature of the network, there needs to be a way for the nodes in the network to reach an agreement among them on the valid transactions that can be appended to the ledger. Consensus protocols are designed to handle faults in a distributed system and agreeing to a single version of the truth by all nodes on the network. The most common types of consensus protocols are leader election based and traditional Byzantine Fault Tolerance (BFT)-based. In a leader election-based consensus, a leader is chosen randomly (by using a protocol) and proposes final valid blocks. The BFT-based consensus is a more traditional method as per rounds of votes. While existing protocols solve the consensus problem fairly well, they also have their own shortcomings. A consensus protocol in a blockchain system is typically required to support three properties: *liveness* (transactions are added to the ledger in a reasonable time), *consistency* (all parties have the same view), and *fairness* (all nodes are equally likely to mine the next block) [13].

Public blockchain networks (e.g. Bitcoin [69] and Ethereum [22]) that use proof

of work (PoW) [69], proof of capacity [39] or proof of activity [17], require a large amount of computational power, which is a misuse of resources and limits transaction throughput (usually expressed as transactions per second). On the other hand, proof of stake [79] and proof of burn[1] [13] are environment-friendly consensus protocols due to the insignificant computation requirements; however, they suffer from the "rich get richer" problem.

Private blockchains, on the other hand, require a centralized party (or a consortium of them) to control who joins the system and at what capacity (mine, view, transact, etc). Such reliance on the centralized party leads to a reduced cost for reaching consensus, high transaction throughput, improved scalability to support new nodes and services, and higher efficiency. In private blockchains, the level of access, visibility, and execution can be controlled. Private blockchains are more appropriate to a consortium of organizations, like the banking sector or the insurance industry, where participation is selective with known identity and may operate under a shared governance model [35]. Examples of private blockchains include Ripple (XRP) [81] and Hyperledger [8]. The Hyperledger Sawtooth project was introduced by Intel as a modular blockchain that uses Proof of Elapsed Time ($PoET$) consensus protocol to implement a leader election lottery system [34, 25]. In $PoET$, each miner node is randomly assigned a `waitTime`, and as soon as this `waitTime` expires, the specific node creates and publishes the next block on the network [25]. The protocol acts as a mix of first-come-first-serve ($FCFS$) and random lottery [87].

Contribution. In this paper, we propose a variant of $PoET$, we call it PoQ, that regulates how nodes compete to finish their `waitTime` such that the average

[1] https://en.bitcoin.it/wiki/Proof_of_burn

wait time and number of leadership each node is assigned is approximately the same across all nodes. Our goal is to optimize the performance of the network concerning *throughput* and *scalability*. PoQ determines which node should execute its `waitTime` when there are multiple run-able nodes in the queue. To achieve this, we introduce the concept of *dynamic Quantum Time* (QT) indicating the amount of time a node will get the chance to execute for a single pass, which has a major impact on resource utilization and overall performance of the network. Some of the extended characteristics that differentiate our work over others include fast transaction processing, low energy consumption, fair distribution, and easy block verification (deterministic). PoQ avoids high resource utilization and replaces it with a true randomized system. Similar to *PoET*, PoQ uses execution environments in trusted hardware, more specifically Intel *SGX* [29], to achieve consensus while preventing tampering. Through the use of Intel SGX enabled CPUs, we enforce correct execution of code and guarantee the "one node one machine" policy (to prevent Sybil attacks) for all nodes in the network. We implemented PoQ in a distributed SGX environment, and our analysis and experiment results show that PoQ provides significant performance improvement over PoET.

4.1.1 Notations

Table 4.1 contains key notations that will be used throughout the chapter.

4.2 Consensus Protocol: PoQ

4.2.1 Overview

This paper introduces a modified version of *PoET* consensus protocol called PoQ based on *SGX*. As part of this protocol, each participating node generates a random

Symbol	Description	Symbol	Description
Q	Queue	EndTime	End Time
$\mathcal{N}$	SGX Node	$\mathcal{ST}$	Starting Time
$\mathcal{E}$	SGX Node enclave	$\mathcal{RT}$	Remaining Time
$\mathcal{S}$	SGX Server	$\mathcal{AET}$	Average Elapsed Time
$\mathcal{SGX}_t$	SGX time	$\mathcal{WT}$	Wait Time
$\mathcal{SGX}_{\min}$	Minimum value of $\mathcal{SGX}_t$	$\mathcal{AWT}$	Average Wait Time
$\mathcal{SGX}_{\max}$	Maximum value of $\mathcal{SGX}_t$	σ	Standard deviation
$\mathcal{SGX}_{\mathrm{T}}$	SGX Table	$\mathcal{Q}_t$	Quantum Time
$\mathcal{N}_n$	Number of active nodes in a specific tier	$\boldsymbol{\mathcal{Q}}_T$	Quantum Time of all nodes
		$\mathcal{P}_k$	Public key
$\mathcal{N}_{id}$	Node id generated by the SGX Server	$\mathcal{S}_k$	Private key
		$\mathcal{N}_{\mathcal{P}_k}$	Node public key
$\mathcal{T}_i$	Tier id for i-th node	$\mathcal{N}_{\mathcal{S}_k}$	Node private (secret) key
$\mathcal{T}_r$	Total number of tiers available. This value is defined by the SGX Server and is uniformly distributed		
${\mathcal{A}_t}^{(i)}$	Arrival Time of i-th node		
$\boldsymbol{\mathcal{A}}_t$	Arrival Times from all nodes		
$\mathcal{ET}$	Elapsed Time		

Table 4.1: Summary of notation used throughout this chapter

`waitTime` using the enclave $\mathcal{E}$, called SGX time $\mathcal{SGX}_t$ and waits for it to be expired. After $\mathcal{SGX}_t$ is finished, the node becomes the leader and is authorized to generate the next block. The `waitTime` and leadership for each node will be approximately the same after a certain period which achieves the equality issue of the consensus protocol.

Random $\mathcal{SGX}_t$. In our protocol, when a node joins the network, it gets a range from the server to generate a random `waitTime`, $\mathcal{SGX}_t$ from its $\mathcal{E}$. After having an $\mathcal{SGX}_t$, the node needs to submit it to the server for further verification. To ensure TEE platforms exists, nodes generally require to register with the hardware manu-

PoQ Server

Initialization. The server $\mathcal{S}$ establishes a public key directory of permitted nodes, creates an empty Q (contains node id) and $\mathcal{SGX}_{\mathrm{T}}$, and then starts listening to requests from nodes interested to join the network.

Node Registration $Register(\mathcal{N}_{\mathcal{P}_k}, Sign(\mathcal{N}_{\mathcal{P}_k}))$. Upon receiving a registration request from a node, the server performs the following:

1. Check whether $\mathcal{N}_{\mathcal{P}_k}$ exists in the public key directory. Otherwise terminate the connection.
2. Check the validity of the signature. Otherwise terminate the connection.
3. Create an identification number, $\mathcal{N}_{id}$.
4. Send an acknowledgment back to the node, along with $\mathcal{N}_{id}$ and $\mathcal{SGX}_{\max}$.

Attestation $Remote\_Att(\mathcal{S}_{\mathcal{P}_k})$.The server and a node jointly execute the RA protocol (Protocol 3.1): Sends a RA request to the client to establish a secure channel.

- The server gets some information from the node which helps to decide whether the program functioning on the node is malicious or fair.

SGX Verification. $Verify(\mathcal{SGX}_t)$. Upon receiving $\mathcal{SGX}_t$, which is a randomly generated time by the node's $\mathcal{E}$ from $\mathcal{N}_{id}$, the server performs the following:

- Check whether $\mathcal{SGX}_t$ is within the range $[\mathcal{SGX}_{\min}, \mathcal{SGX}_{\max}]$. Otherwise terminate the connection.
- Add $\mathcal{N}_{id}$ to Q and build $\mathcal{SGX}_{\mathrm{T}}$ that has $\mathcal{Q}_t$ and $\mathcal{ST}$ for all nodes based on available information.
- Send meta-data ($\mathcal{N}_n$, $\mathcal{N}_{\mathcal{A}_t}$, $\mathcal{N}_{\boldsymbol{\mathcal{Q}}_T}$, $\mathcal{T}_r$, $\mathcal{N}_{\mathcal{RT}}$) to $\mathcal{N}_{id}$.

Status. Server will perform the following operation in meta-data:

- Continuously updates the $\mathcal{SGX}_{\mathrm{T}}$, queue and dequeue the winner nodes to keep a track.
- Broadcast the result to the network.

Protocol 4.1: PoQ Server side protocol.

facturer to set up RA services. For instance, Intel SGX RA service needs registration with Intel Attestation Service (IAS)[2]. During manufacturing, each processor of SGX

[2]Intel, "Software sealing policies – intel® software guard extensions developer guide," 2017. [Online]. Available: https://software.intel.com/en-us/documentation/sgx-developer-guide

PoQ Node

1. The node signs its public key and then sends a node registration request $Register(\mathcal{N}_{\mathcal{P}_k}, Sign(\mathcal{N}_{\mathcal{P}_k}))$ to $\mathcal{S}$. Upon successful registration, the node receives an acknowledgment with its $\mathcal{N}_{id}$ and $\mathcal{SGX}_{\max}$.
2. Initialized by the server, the node jointly executes the $Remote\_Att(\mathcal{S}_{\mathcal{P}_k})$ (Protocol 3.1) with the server to ensures its identity and it is running on an Intel SGX enabled platform without tampering.
3. The node performs the following steps in order to participate into the PoQ protocol:
   (a) Generate a random $\mathcal{SGX}_t$ from $\mathcal{E}$ within a range of $[\mathcal{SGX}_{\min}, \mathcal{SGX}_{\max}]$.
   (b) Broadcast and request $Verify(\mathcal{SGX}_t)$ to $\mathcal{S}$, hence gets meta-data that states information about the existing nodes in the network.
   (c) Determine tier $\mathcal{T}_i$ it belongs to according to the following:
   $$\mathcal{T}_i = \left\lceil \mathcal{T}_r \frac{\mathcal{SGX}_t}{\mathcal{SGX}_{\max}} \right\rceil \tag{4.1}$$
   (d) Calculate the local $\mathcal{Q}_t$ using the following formula:
   $$\mathcal{Q}_t = \left\lceil \frac{\sum_{i=1}^{\mathcal{N}_n} \mathcal{RT}}{{\mathcal{N}_n}^2} \right\rceil \tag{4.2}$$
   where $\mathcal{N}_n$ is the number of active nodes in the specific $\mathcal{T}_i$.
   (e) Obtain $\mathcal{ST}$ using the information from $\mathcal{SGX}_{\mathrm{T}}$.
4. Once it gets $\mathcal{ST}$, $\mathcal{SGX}_t$ will be reduced for the calculated $\mathcal{Q}_t$.
   While Remaining Time $\mathcal{RT} \neq 0$, then:
   (a) Generate a new $\mathcal{Q}_t$ and determine the next $\mathcal{ST}$.
   otherwise:
      i. A new block is propagated and the local leadership count is incremented by one.
      ii. Broadcast the winning result to $\mathcal{S}$ and announce new block.
5. To rejoin the network, steps 2 to 4 are repeated.

Protocol 4.2: Individual node side protocol.

is equipped with a key that is certified by Intel [5]. After successful verification, the server adds the node id, $\mathcal{N}_{id}$ of that node to the queue, Q as it arrives. Subsequently,

the node determines which tier, $\mathcal{T}_i$ it belongs to and then calculates its $\mathcal{Q}_t$ for that particular $\mathcal{T}_i$ for that time being which is equal to the amount of time it can be executed for its first pass. If it remains in the waiting part of the Q and any node joins which belong to its $\mathcal{T}_i$ then it needs to recalculate the $\mathcal{Q}_t$ again based on available data. A new node can also be added at the end of the Q. While a node is executing and a new node joins that belongs to the same $\mathcal{T}_i$ it won't affect the $\mathcal{Q}_t$ of the executing node at that moment. However, if the node is unable to finish the entire $\mathcal{SGX}_t$ during that pass of $\mathcal{Q}_t$, it will be popped up from the Q and added again at the end of it without changing its $\mathcal{T}_i$ but this time it needs to recalculate the $\mathcal{Q}_t$ for its next pass. Then, the node who is in the starting point (starting time, $\mathcal{ST}$ = current time) of the Q will get the chance to reduce its $\mathcal{SGX}_t$. After completion of each node's $\mathcal{Q}_t$, the remaining $\mathcal{SGX}_t$ of the currently executing node is checked. A function "Time Left" keeps track of the *Remaining Time* ($\mathcal{RT}$) over $\mathcal{SGX}_t$ after each pass and once it has zero as its value, it will broadcast the result for claiming the leadership. A participating node is required to finish all its $\mathcal{SGX}_t$ to become the leader and propagates a new block. It is worth mentioning that, the total waiting time of a node is not equal to its $\mathcal{SGX}_t$.

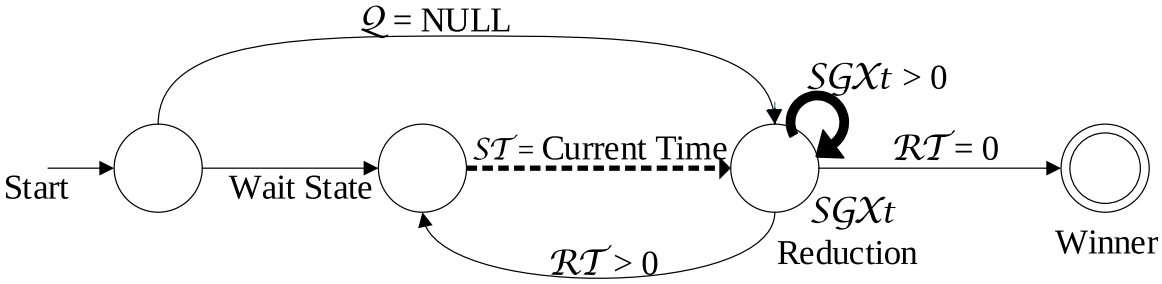


Figure 4.1: The various state of a node before becoming a leader.

4.2.2 Principals:

Our protocol consists of two phases: server and client. The initial step is registration. In this phase, nodes need to join the network for authentication. Nodes are the major principals in the PoQ consensus protocol and rely on TEE. TEE can generate an independent identical random digit, which cannot be controlled by an advisory. RDRAND command is available in Intel SGX. Our design uses minimal energy consumption and exploits the Intel SGX floor.

4.2.3 Protocols:

In the initial phase, Intel SGX plays a crucial part. Our protocol develops an information flow between a local SGX and the server proposed in protocol 1 and 2.

Protocol 4.1 - Server side protocol. The server will always wait for a join request from nodes, $\mathcal{N}$. Whenever it gets a request, it calls $Register(\mathcal{N}_{\mathcal{P}_k}, Sign(\mathcal{N}_{\mathcal{P}_k}))$ method where it verifies the authentication with the directory of permitted public keys, $\mathcal{N}_{\mathcal{P}_k}$. If it is valid, the server generates an identification number, $\mathcal{N}_{id}$, inserts it into the Q and sends an acknowledgment to the newly joined node with the range of the $\mathcal{SGX}_t$ and $\mathcal{N}_{id}$. Immediately after receiving $\mathcal{SGX}_t$ from an node, $\mathcal{N}_1'$, it calls the $Verify(\mathcal{SGX}_t)$ method where it checks the $\mathcal{SGX}_{t1}'$. If it does not fit in the range, it aborts the connection. Thus, no node can go after the smallest number that is beyond the range to generate too many blocks. After successful verification of $\mathcal{SGX}_{t1}'$, the server stores the time when it's submitted and treats it as its arrival time, $\mathcal{A}_{t1}'$ of $\mathcal{N}_1'$. Then calculates $\mathcal{Q}_{t1}'$ according to its tier $\mathcal{T}_i$ and obtains the first starting time, $\mathcal{ST}_1'$, of that particular node $\mathcal{N}_1'$. Then the server builds an SGX Table, $\mathcal{SGX}_{\mathrm{T}}$ based on available data that contains the number of nodes, $\mathcal{N}_n$, arrival time $\mathcal{N}_{\mathcal{A}_t}$, quantum time, $\mathcal{N}_{\mathcal{Q}_T}$, and remaining $\mathcal{SGX}_t$ for all active nodes $\mathcal{N}_{\mathcal{RT}}$ with numbers of tiers.

The server sends this meta-data to all active nodes. Thus, all $\mathcal{N}_{id}$ have access to a similar database concurrently that server has, and almost every data will replicate to all $\mathcal{N}$ which accelerates the speed of the network. The server always monitors the scheme of finding a new leader. When a new leadership has been claimed, a node $\mathcal{N}_1'$, is supposed to announce that it has completed its $\mathcal{SGX}_{t1}'$, so that it is considered as a leader and get appended to the blockchain and dequeued from the Q. If that particular node wants to rejoin in the network its $\mathcal{N}_{id}$ will remain the same with different credentials. Whenever a new node, $\mathcal{N}_2'$, join or a node $\mathcal{N}_1'$ leaves, the server continuously updates the Q based on all available data. However, the server can compute Average Elapsed Time, $\mathcal{AET}$ and Average Waiting Time, $\mathcal{AWT}$ by the following formulas:

$$\mathcal{AET} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{ET}_i \tag{4.3}$$

$$\mathcal{AWT} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{WT}_i \tag{4.4}$$

Definition 1 *Waiting Time.* The inactive time of a $\mathcal{N}_i$ after consider its $\mathcal{A}_{ti}$. Simply, the amount of idle time $\mathcal{WT}_i$ spent by a $\mathcal{N}_i$ in the Q before the last pass to finishes its $\mathcal{SGX}_{ti}$ for a single round can be calculated by Eq (4.6); where $\mathcal{A}_{ti}$ refers to $\mathcal{A}_t$ of $\mathcal{N}_i$ and $\mathcal{SGX}_{ti}$ refers to the time generated from $\mathcal{E}$ of $\mathcal{N}_i$. $\mathcal{AWT}$ is the average value of wait time of N nodes and can be calculated by Eq.(4.4).

Definition 2 *Elapsed Time.* The entire time requires a $\mathcal{N}_i$ to become a leader. That means the time elapsed between $\mathcal{A}_{ti}$ of a $\mathcal{N}_i$ and its termination. Elapsed time for a $\mathcal{N}_i$ can be calculated by Eq. (4.5); where $\mathcal{WT}_i$ refers to $\mathcal{WT}$ of $\mathcal{N}_i$ and $\mathcal{SGX}_{ti}$ refers

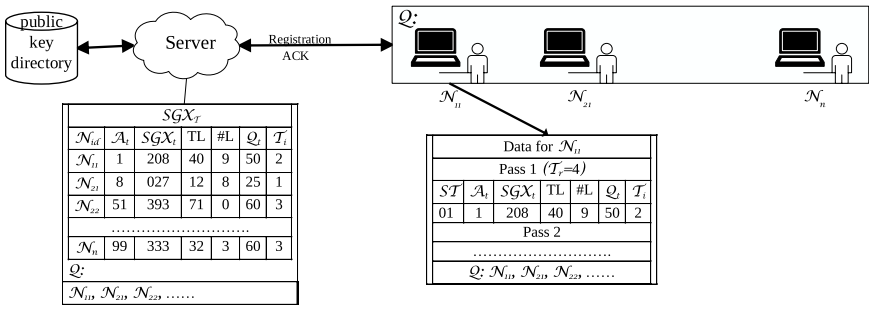


Figure 4.2: Top level architectural diagram of the system.

to the time generated from $\mathcal{E}$ of that $\mathcal{N}_i$. The average elapsed time of N nodes $\mathcal{AET}$ can be calculated by using Eq. (4.3).

Protocol 4.2 - Individual node side protocol. At first, a participant node needs to register for joining the network. After joining, a local SGX acts as a client node that requires to download the PoQ code and execute it. When a local SGX connects to the server, it gets $\mathcal{N}_{id}$ and the range to generate a random $\mathcal{SGX}_t$ (which is subject to change subsequently after each round) from the trusted code inside $\mathcal{E}$ and needs to submit it to the server for verification, which is done on the same platform. If there is more than one node ($\mathcal{N}_2'$, $\mathcal{N}_{21}''$) who produces $\mathcal{SGX}_t$ at the same time then they will be added into the Q as ascending order of $\mathcal{N}_{id}$. If Q is null then $\mathcal{N}_{id}$ is added at the front of Q. If Q is not null but the $\mathcal{RT}$ of the current executing node is zero then $\mathcal{N}_{id}$ is added in $FIFO$ manner, otherwise, $\mathcal{N}_{id}$ will insert into the Q after the $\mathcal{N}_{id}$ of the executing node. Later, the node gets the $\mathcal{SGX}_{\mathrm{T}}$ from the server that consists of its $\mathcal{Q}_t$, $\mathcal{ST}$ and $\mathcal{A}_t$ along with some meta-data ($\mathcal{N}_n$, $\mathcal{N}_{\mathcal{A}_t}$, $\mathcal{N}_{\mathcal{Q}_T}$, $\mathcal{T}_r$, $\mathcal{N}_{\mathcal{RT}}$). If we consider there are only three nodes ($\mathcal{N}_2'$, $\mathcal{N}_{21}''$, $\mathcal{N}_3'$) in the network where $\mathcal{N}_2'$, $\mathcal{N}_{21}''$ has same $\mathcal{A}_t$ and $\mathcal{N}_3'$ join after two units of time, then the start time of $\mathcal{N}_2'$ which is $\mathcal{ST}'(\mathcal{N}_2')$ immediately when it arrives and $\mathcal{ST}''(\mathcal{N}_{21}'')$ is after the amount of quantum time of $\mathcal{N}_2'$. The $\mathcal{ST}'(\mathcal{N}_3')$ is the addition of all the nodes $\mathcal{Q}_t$ left to execute, who are

in front of it in the Q at the moment it arrives. Based on all available information, a particular node, $\mathcal{N}'$, can also calculate which $\mathcal{T}_i$ it belongs to, $\mathcal{Q}_t{}'$ of that specific $\mathcal{T}_i$ and $\mathcal{ST}'$. For a specific tier whoever comes first will start first. When a new $\mathcal{N}$ is added to a particular $\mathcal{T}_i$, the $\mathcal{Q}_t{}'$ of that $\mathcal{T}_i$ is recalculated according to the available updated information based on $\mathcal{RT}$ of all nodes in that $\mathcal{T}_i$. When $\mathcal{ST}$ is equivalent to the current time, nodes will execute to reduce its $\mathcal{RT}$. When a node joins, the amount of its $\mathcal{RT}$ is equivalent to its $\mathcal{SGX}_t$.

After a single pass, if a particular node, $\mathcal{N}_1'$, is not able to finish its $\mathcal{SGX}_{t1}{}'$ as a whole, $\mathcal{RT}_1'$ will be updated by deducting the time spent on that pass and it will put itself at the end of the updated queue, then calculates its next $\mathcal{ST}_1''$ and needs to wait for another pass. If there is no $\mathcal{N}$ in the overall Q than the current node may carry on. It is mentioned that, at any stage, for a particular $\mathcal{N}_1'$, if $\mathcal{SGX}_{t1}{}'$ or $\mathcal{RT}'_n$ is less than the $\mathcal{Q}_{t\mathcal{T}_i}'$ then the $\mathcal{Q}_{t1}'$ will be updated and assigned to the equal portion of that specific $\mathcal{RT}'_n$. If any node finishes its $\mathcal{SGX}_t$, then it will be withdrawn from the Q and becomes the leader and the number of leadership is assigned to it will be increased by one. Thus, a new block is propagated. Figure 4.2 elucidates the top-level architecture of PoQ and figure 4.3 shows the inter-process communication between the nodes and server. However, the node can compute its Elapsed Time and Wait Time by the following formulas:

$$\mathcal{ET}_i = \mathcal{SGX}_{ti} + \mathcal{WT}_i \tag{4.5}$$

$$\mathcal{WT}_i = \text{EndTime}_i - (\mathcal{SGX}_{ti} + \mathcal{A}_{ti}) \tag{4.6}$$

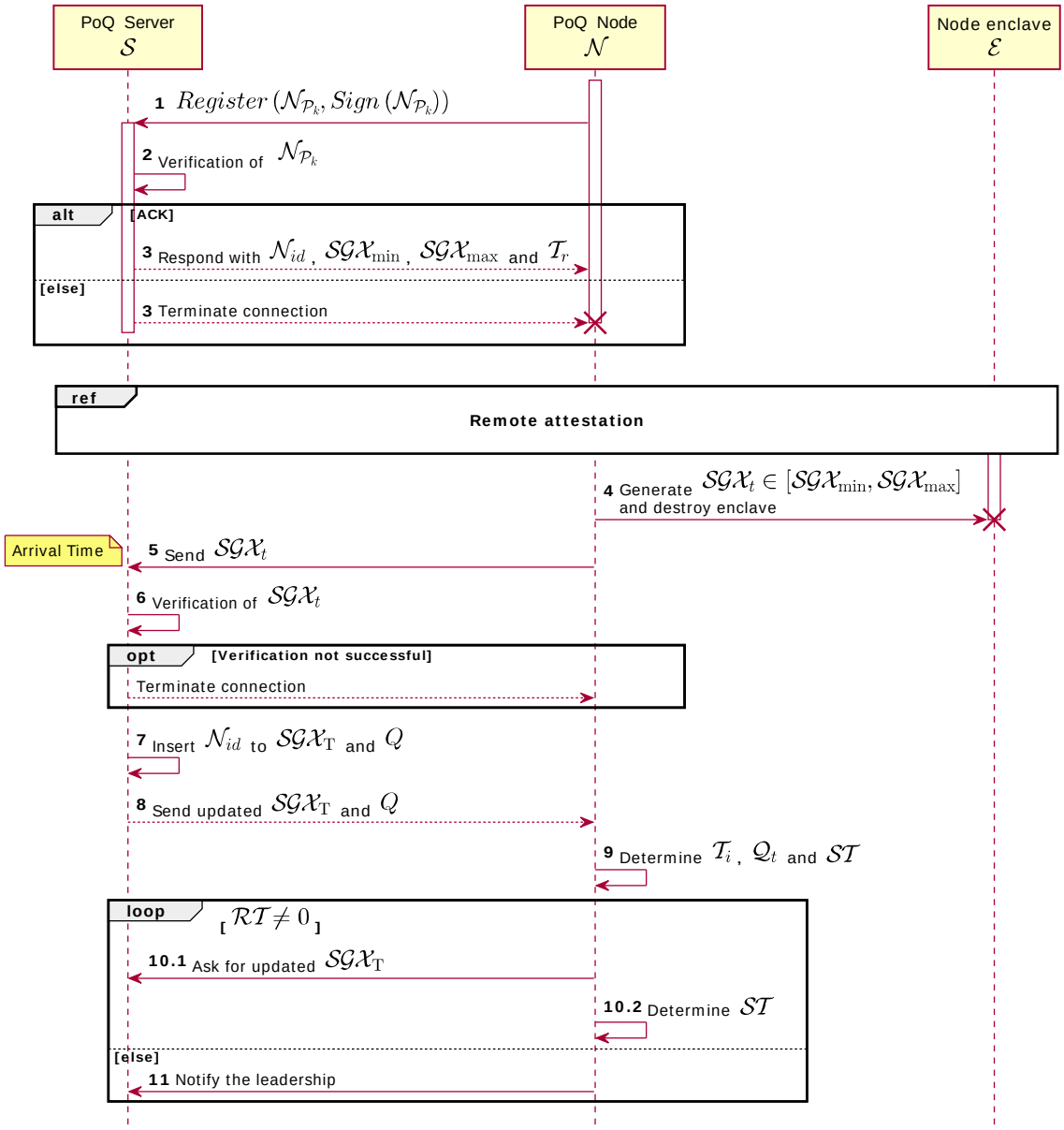


Figure 4.3: Interactions corresponding to client-server communication in PoQ.

4.3 Experimental Evaluation

4.3.1 Goals

The design of a good consensus protocol must satisfy the following goals: (i) backing up a large-scale network (ii) obtain a higher throughput, and (iii) achieve fairness.

To better motivate and illustrate our design, we performs these experiments to achieve those goals throughout the experiments.

4.3.2 Setup

We built a prototype of PoQ to evaluate its performance. All practical experiments performed below were done using a system equipped with SGX PSW 2.X of version 2.5.100.2 and SGX SDK 2.X of version 2.5.100.2 which acts as a in-house client-server network. The system has Windows 10 OS with the latest updates, Intel® Core™ i7-7567U processor (3.5 GHz to 4.0 GHz Turbo, Dual Core 4 MB cache, 28W TDP), 32GB RAM, 64 Mb Flash EEPROM, and 34.1 GB/s Max Memory Bandwidth[3]. We assume that every full node is a potential validators.

4.3.3 Throughput

The purpose of this experiment is to measure and compare the throughput of PoQ and PoET. We ran multiple experiments with different parameters. We measure the number of leaderships per second for ten different nodes and three $\mathcal{SGX}_t$ ranges: [1,100], [1,500], and [1,1000]. In the baseline case, we assume that all nodes arrive approximately at the same time: within the first two (Figure 4.4.a), ten (Figure 4.4.c), and twenty (Figure 4.4.e) seconds . Then, we allow those ten nodes to join randomly at different times within a certain range of $\mathcal{A}_t$ with the same $\mathcal{T}_r$ which is 5. We ran each test 50 times where $\mathcal{SGX}_t$ and $\mathcal{A}_t$ were generated randomly. The time duration for each run for Figure 4.4.a and 4.4.b were 90 sec ($\mathcal{A}_t \in$ [0,300]s), 450 sec ($\mathcal{A}_t \in$[0,1500]s) for Figure 4.4.c and 4.4.d, and 900 sec ($\mathcal{A}_t \in$[0,3000]s) for Figure 4.4.e and 4.4.f. In Figure 4.4, the graph with different arrival times deals with the average

[3]Max Memory Bandwith is the maximum rate at which data can be read from or stored into a semiconductor memory by the processor (in GB/s).

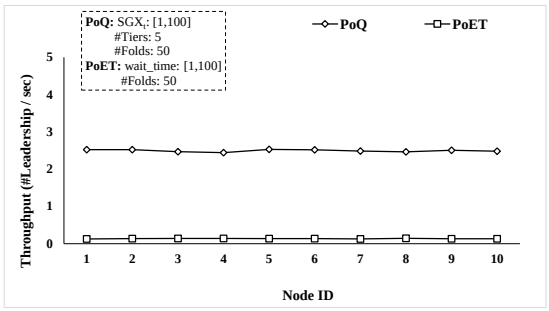
result of tests where 20% of nodes leave the network randomly at any time after becoming a leader at least once. Note that the protocol was slightly modified when performed 20% nodes left. For baseline case, experiments were run with the same settings as discussed above and the final result is averaged where node does not leave the network. For comparison, we implement PoET and run the same experiments with the same attributes to evaluate the performance with respect to PoQ.

By comparing PoQ with PoET in Figure 4.4, we observe that the throughput of PoQ is higher in both cases: all nodes join approximately at the same time (baseline case), and when they join at different times. The difference between the two protocols' throughput could be as low as 0.3 (Figure 4.4.2) and as high as 3.5 (Figure 4.4.d).

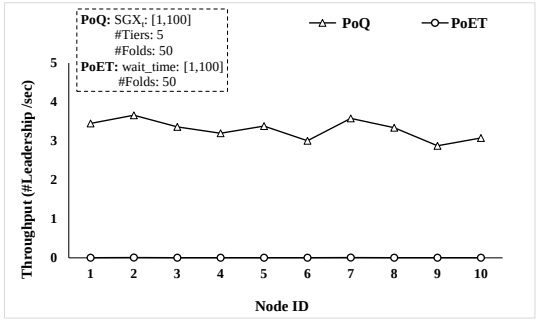
4.3.4 Scalability

In this section, we evaluate the scalability of our protocol. We start with a network of 2000 nodes then double the network size 5 times, raising to 10,000 nodes in the last setting. In Figure 6.3 we keep the same parameters involved in Figure 4.4.a but with a larger number of nodes. We ran the experiment only once until all the nodes become exactly one leader. It should be noted that epoch time is longer (e.g., 100 seconds in 2000 nodes to 504 seconds in 10,000 nodes) since it requires relatively more times when the total number of nodes increases.

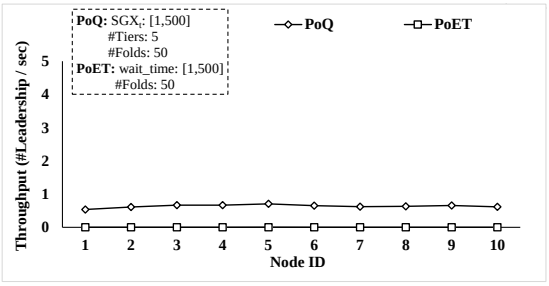
By observing the graph we conclude that $\mathcal{AWT}$, $\mathcal{AET}$ for different sizes of network increase linearly as the network size grows. We also measured RA which takes roughly 2ms and we did not consider it in result data.



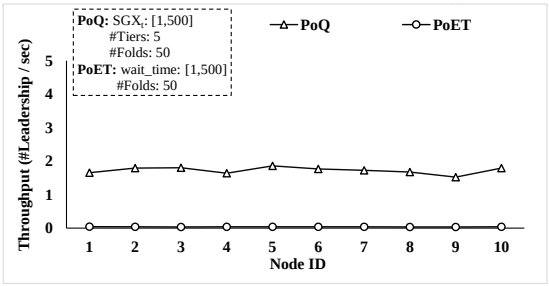
(a) Nodes start approximately at the same time, $\mathcal{A}_t \in \{0,2\}s$



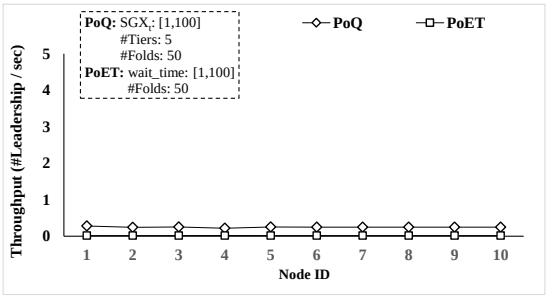
(b) Nodes start at different time, $\mathcal{A}_t \in \{0,300\}s$



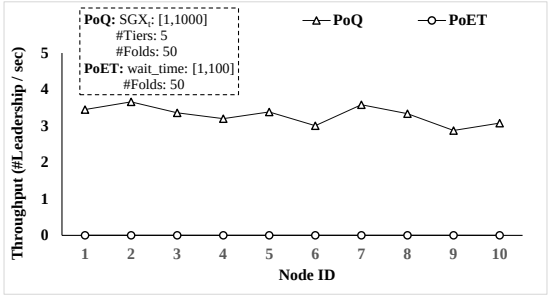
(c) Nodes start approximately at the same time, $\mathcal{A}_t \in \{0,10\}s$



(d) Nodes start at different time, $\mathcal{A}_t \in \{0,1500\}s$



(e) Nodes start approximately at the same time, $\mathcal{A}_t \in \{0,20\}s$



(f) Nodes start at different time, $\mathcal{A}_t \in \{0,3000\}s$

Figure 4.4: Throughput evaluation results among ten nodes for PoQ and PoET. Each data point in our plots is averaged over 50 independents measurements.

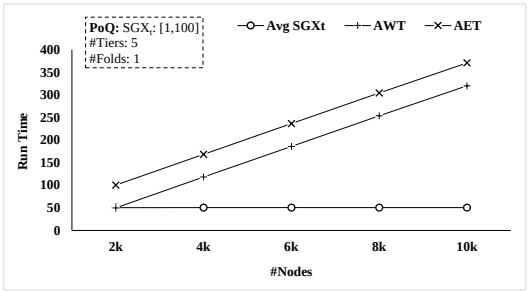


Figure 4.5: Overview of bridging between $\mathcal{AWT}$ and $\mathcal{AET}$ in PoQ in response to scalability (up to ten thousands of nodes).

4.3.5 Fairness

A consensus protocol is fair if a miner/validator with p share of the overall resource ratio can produce a block with a probability p. In this section, we trying to measure the relation between the number of SGX machines a node has and the number of leadership it can reach. We conducted this experiment with the same parameter elaborated in Figure 4.4.a. We ran the experiment 50 times and the Figure 6.2 reported below are averaged over fifty independent runs. The graph shows the cumulative average leadership of a validator who has a certain number of SGX per node. The X-axis indicates the total number of SGX a node has and the Y-axis shows the average leadership.

After running our experiments, described above, we observe that the probability of being chosen to be a leader scaled linearly, in relation to the number of SGX machines per node.

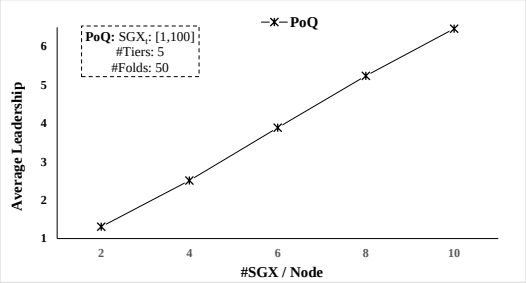


Figure 4.6: A linear growth in experimentation over nodes.

4.4 Conclusions

In this work, we proposed Proof of Queue (PoQ), a leader based consensus protocol for private (permissioned) blockchains that utilizes Intel SGX to ensure all nodes in the system honestly run trusted code to become a leader. PoQ maintains queues

for different tiers, keeps tracks of the quantum times executed by all nodes, and updates the state changes to all nodes. PoQ is specifically designed to avoid the collision in leader election than existing PoET protocol. Also, our protocol is suitable for a large number of nodes with an enormous wait time. The design of PoQ shows that it maintains approximately similar wait times and elapsed times for all the nodes. Finally, based on the simulation and large-scale evaluation of PoQ, we showed that no nodes are left behind. We evaluate PoQ against three metrics; throughput, scalability, and fairness. The results show that PoQ can offer enhanced scalability. Besides, PoQ scales fairness linearly with SGX machines.

CHAPTER 5:

ACCORD: A SCALABLE QUORUM-BASED CONSENSUS PROTOCOL FOR HEALTHCARE BLOCKCHAIN

5.1 Introduction

In Electronic Health Record (EHR) systems, all health-related records are digitized and independently stored in the hospital's local database. However, a patient may visit more than one medical institution for different needs or may be transferred from one institution to another. The procedure of forwarding the data is often difficult and time consuming. A survey in 2014 found that 15% of patients who visited a healthcare provider in the US reported having to bring test results to their appointment personally, and 5% needed to have a test or procedure repeated due to the unavailability of prior results [74]. According to the US ONC Health IT[1], in 2019, roughly half of US hospitals were considered interoperable, meaning they are able to efficiently transfer medical records between each other. Even if those concerns could be addressed, the security of health data needs to be considered as they are primary targets to cyber attackers. Secure storage of patients' health data is

[1]https://www.healthit.gov/sites/default/files/page/2021-02/Use-of-Certified-Health-IT-and-Methods-to-Enable-Interoperability-by-U.S.-Non-Federal-Acute-Care-Hospitals-2019.pdf

paramount in healthcare. This data must, by ethics and law, be kept private. This, compounded with the value of this data being far greater than financial information like credit cards to cyber-attackers [4], requires a secure method to store and access this data [18]. The importance of EHRs implies that they should not be consolidated in one place where they can be attacked. Many existing EHR systems use Distributed Database Management System (DDBMS), which has higher throughput and lower latency than standard blockchain solutions but can be lacking in other areas, including interoperability between hospitals and patients control of medical records. Table 5.1 shows a feasibility study of blockchain superiority over DDBMS.

Hospitals have many methods of storing medical information. According to the CDC, in 2017, approximately 85% of hospitals use some form of EHR[2]. There is no standard for EHR systems, so these systems are widely varied and manage data in significantly different ways [1]. This leads to patient confusion as to where their data is and how to access it [23]. EHRs are distributed among different medical institution's local databases. The sharing of EHRs over multiple institutions is a complicated process [36]. This results in poor communication between hospitals. From the patient's point of view, this leads to repeated tests and procedures [45, 77, 88], which not only inflates their medical bills but also cause negative patient outcomes. However, in a blockchain-based platform, patient records can show up as a single list of consecutive care events, regardless of where these events occurred [76]. In March 2020, the US Department of Health and Human Services (HHS) created a policy such that patients should be given control of their personal health data [48]. Therefore, we see it as desirable for the patient to have direct access to their EHRs.

[2]https://www.cdc.gov/nchs/data/nehrs/2017_NEHRS_Web_Table_EHR_ State.pdf

With the help of blockchain technology, patients can be given control of their health data. Blockchain technology can potentially realize these goals in the healthcare sector to create a patient-centric system for EHR management.

For a blockchain network to be functional and secure, its nodes (i.e., hospitals) must not conflict with the distributed ledger's current state. This is achieved through a consensus protocol. Consensus protocols are responsible for ensuring that the nodes in the network agree that collections of records (called *blocks*) are valid and appropriately added to the distributed ledger. These blocks of digital records are immutable and do not require trust [71]. Note that these records can be encrypted and anonymized to maintain patient privacy. A secure consensus protocol allows hospitals to be assured that their ledger is identical to other the hospitals' ledger, preventing omissions and tampering. As there are around 9,000 hospitals and clinics in the United States[3] and as many as 30 million transactions created per day [2], the speed and efficiency of the consensus protocol are important in a healthcare blockchain.

Two of the most common consensus protocols in blockchain are Proof of Work (PoW) and Proof of Stake (PoS). PoW [69] is the consensus protocol used by the seminal work in blockchain. It allows for a permissionless mining system, which is desirable in cryptocurrency. A miner can propose a block if they can solve a hard puzzle, and the first miner to solve the puzzle associated with their block gets a reward. This results in a processing power lottery, where whoever solves the puzzle first gets the reward for solving the block.

However, this results in the PoW being an extremely energy inefficient consensus protocol. It demands that miners purchase powerful hardware to have a meaningful

[3]https://www.aha.org/statistics/fast-facts-us-hospitals

impact. Also, security is an issue with PoW. Bitcoin, which uses PoW, is vulnerable if only 25% of the computing power is controlled by an adversary [86]. Alternatively, in PoS, there is no need to solve a hard puzzle, as validators can validate the next block. A miner's ability to mine a block is proportional to the stake they have in the system. While this is more energy-efficient than PoW, PoS depends on the miners having a stake. In a cryptocurrency setting, a participant's stake can be determined or purchased with the currency being exchanged. In other contexts, the determination stake becomes unclear.

These consensus protocols allow any party to mine, which is not necessarily required or desired in a healthcare blockchain.

An alternative paradigm is where the miners must be known and approved to mine. A popular consensus in this paradigm that has been used in the healthcare sector is Practical Byzantine Fault Tolerance (PBFT) [84]. Currently, a version of PBFT is using in Hyperledger Fabric [47]. PBFT works efficiently for a small network size. However, due to the high communication overhead, PBFT does not scale well. This is because each node must communicate with all other nodes at every step to keep the network secure. As the amount of nodes scales upwards (increases as $O(m^n)$, where m is the messages and n is the number of nodes), the communication burden of this system becomes untenable. The system can continue functioning properly with up to 33% corruption of the mining network. Delegated Proof of Stake (DPoS) deals with a voting system where stakeholders outsource their work to a third-party. Essentially, nodes can outsource their stake to a delegate who will propose blocks and validate blocks. The voting power is proportional to the amount of stake that each validator holds. Due to the centralization of mining power, DPoS is not suitable

for a healthcare blockchain [40]. According to the Hasselgren et al. survey [47], the most commonly used consensus protocols in healthcare publications are PoW (21%) and PBFT (15%). Unfortunately, this survey found that 41% did not explicitly state which consensus protocol they use or recommend for their system.

Petersen et al. created a consensus protocol for a healthcare blockchain [76]. It attempts to create a consensus by having all participating nodes agree on a block with a coordinator. This system's miner selection process, however, leaves much to be desired. It requires a full network synchronization between each block to select the next miner. This is not only inefficient as the network grows, but it also opens the miner selection process to malicious action. If a malicious or malfunctioning miner refuses to send their seed or sends it to only a subset of miners as the selection closes, it could either stall the network or cause the network to disagree over who the next miner is. This means that the system lacks robustness. Additionally, the block creation process is open to the leader omitting transactions. If a hospital does not send the elected miner any transactions, they are not required to sign or approve the block. This could result in the miner ignoring a hospital's transactions and claiming they never received any messages from them, which allows the miner to omit that hospital's transactions. Requiring each node to participate in the production of a block required a large amount of communication that scales with the number of hospitals in the network. If a hospital is offline or malicious, they may not sign the block, which could delay block creation and result in another expensive round of communications to create another block. Also, linking a transaction to a hospital can reduce the privacy of a transaction, making it more probable that a transaction can be linked to a patient.

In MedBlock [40], Fan et al. designed a system that attempts to solve the communication difficulties of PBFT with large network sizes. They created a form of delegated PBFT, where hospitals are divided into regions. Each region elects a representative to represent them in the larger network. These representatives then execute PBFT among themselves. This aims to reduce network congestion and the higher energy costs this congestion entails. However, this opens smaller clinics or hospitals to malicious action. If the majority of a region wishes to suppress a node, there is no obvious recourse within the protocol other than manual intervention (i.e., moving the affected nodes out of the region or suspending the malicious nodes).

The current medical data infrastructure mostly depends on trusted third parties. A public permissioned blockchain is a potential solution to this issue. A blockchain can provide privacy, security, accessibility, and reliability of sensitive data. A blockchain can also create an immutable chain of patient care events, regardless of where these actions were carried out. These features imply the most acceptable type of blockchain in healthcare would be a public permissioned blockchain.

We define a public permissioned blockchain to be a blockchain in which the ledger is public, but the consensus protocol is permissioned. This allows for public verifiability while still allowing for a more efficient consensus mechanism. In our context, permission to mine is granted by the Membership Service Authority (MSA). The MSA will grant permission to a party that has the credentials to join the mining pool.

Picking an appropriate consensus protocol is important when developing a practical blockchain that is operable in the healthcare sector. There is a larger number of potential miners that may be involved in the protocol than in many other systems.

Existing protocols may be appropriate for smaller systems, but may not provide the features required for a medical system. Characteristics to identify the appropriate consensus protocol for a permissioned blockchain include the following: fork resistance, proportionate mining, transaction fairness, energy efficiency, and fault tolerance.

In this work, we propose a salable consensus protocol, named ACCORD that avoids conflicts over forks and can withstand network outages. The protocol works as follows: After a transaction is forwarded to the peer-to-peer network, ACCORD utilizes a quorum-based consensus protocol to evenly distribute the responsibilities of a single leader to multiple quorum members. We ensure the correctness of the block by a threshold of the quorum members agreeing on the transactions before proposing their block. For this block to be accepted by the network, it must be asynchronously signed by a majority of the nodes in the network before it is added to the blockchain. Figure 5.1 is a high level view of how data is processed by ACCORD. The main characteristics of the ACCORD protocol are as follows:

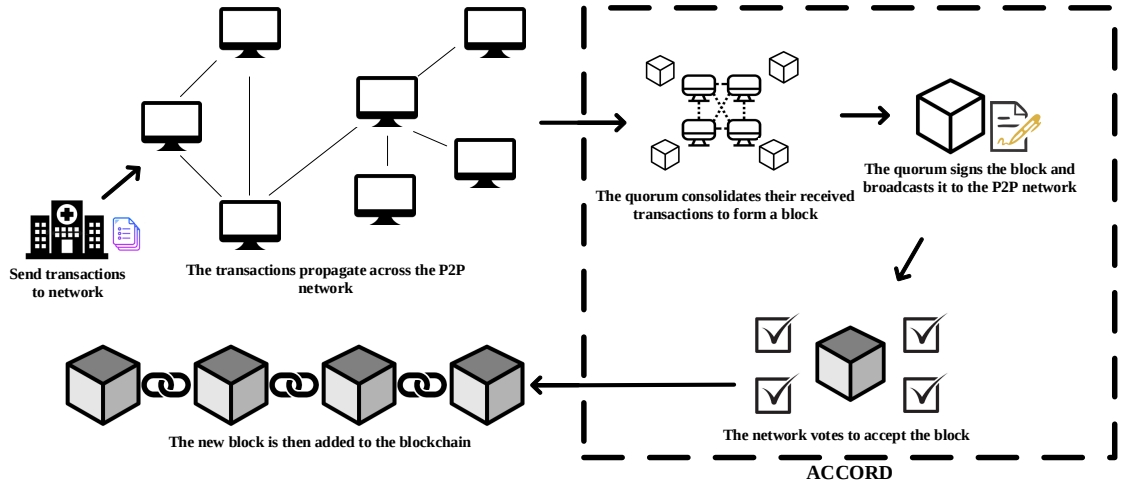


Figure 5.1: A high level architectural diagram of the system, showing the path a transaction takes to be added to the blockchain.

- Fork-Resistance[4]: In honest execution, our protocol will not create forks. In malicious execution, either one path is chosen that the network asynchronously agrees on before future blocks are created or all the existing forks are abandoned and a new quorum is selected to mine a replacement branch.

- Robustness: In the event of a network outage, our protocol can remain functional or recoverable as long as more than 50% of the network's total authorized nodes are both honest and online.

- Scalability: Our protocol scaled linearly as it continues to produce blocks with an increasing number of authorized mining nodes and transactions. Additionally, an unbounded block size can result in an unlimited number of transactions per block.

- Liveness and safety: Our consensus protocol offers the liveness and safety property. A network that affirms *liveness* ensures a valid transaction will appear in all honest node's ledgers within a reasonable period.

In addition, ACCORD achieves fairness in miner selection and fairness in transaction addition. ACCORD is fair in miner selection because it ensures every miner is selected with approximately the same frequency with a more even distribution than random selection. ACCORD achieves fairness in transaction addition by requiring that every honest node adds every transaction they are aware of to the block and by ensuring there are regularly blocks that behave honestly.

[4]Fork resistance is not immunity to forks, where forks cannot occur. In fork-resistance, forks can occur, but if one honest node accepts one fork, another honest node will not accept another.

Table 5.1: Feasibility study of blockchain over DDBMS in healthcare.

Decentralized and Accessibility	Patient data should belong to patients and it should not be controlled by any central authority. Patients should be able to view their records and easily grant access to another provider.
Immutability, Privacy and Security	In a DDBMS, an admin can execute four functions on data: Create, Read, Update, and Delete (known as the CRUD commands). On the other hand, traditional blockchains are generally append-only structures, which means that users can execute only create and read operations. This kind of inflexibility gives security and permanence to EHRs.
Data provenance and Misuse	Data provenance allows for individual items (e.g. tests, treatments) to be taken in a larger context, which can be used to find errors in medical care. A blockchain can maintain a complete history of the state of a patient's health records, where an admin in DDBMS can alter or delete the data. Database admins require a high level of trust that they will conduct their roles correctly, whereas a blockchain requires a lower level of trust that can be distributed across a large group.
Robustness and Durability	Blockchain has built-in robustness and durability functionality. It does not suffer from a single point of failure because a set of nodes interact towards keeping a secured ledger, thus avoiding the issues that come with the data being stored in a central service. This functionality becomes expensive when it comes to DDBMS.

5.2 Adversary model

In our consensus protocol, ACCORD, there are five major groups of parties: the transaction makers (e.g. doctors, hospitals), the mining nodes, the membership authority, the peer-to-peer nodes, and external observers.

We assume that the transaction makers create the transactions honestly and do not distribute any private data. However, we also assume that they are willing to modify the data after the fact if given the opportunity. An example of this would be a doctor attempting to alter lab results on the blockchain to avoid a malpractice lawsuit.

We assume that the mining nodes are covert, meaning they are willing to violate the protocol but are not willing to get caught. If they can drop transactions, cause a slowdown, or gain leverage over the system while avoiding detection, they are willing to do so. However, we assume they are malicious if they are capable of damaging the network. Damage to the network includes the following: violating safety by convincing separate honest nodes of conflicting states in the blockchain, extended outages of the network, and extended periods of control over the network. Unless they are able to cause damage to the network, they are unwilling to receive any off-chain punishment for malicious action. Due to our quorum selection protocol (discussed in algorithm 1), we also assume they are able to create a block that produces the exact results that they want from the quorum selection protocol without detection if they have control over the block creation (i.e. a malicious quorum can select specific future miners). We also assume that a simple majority of the miners will behave honestly.

We assume the membership service authority (MSA) is semi-honest. They act as a certificate authority and are not privy to any private data on the blockchain.

They will only authorize new miners if they are qualified. Note that one of our recommended options is that the membership service authority is the whole mining pool, able to act with a simple majority vote. This option leads to the semi-honesty of the membership authority being reduced to the fact that a simple majority of the mining nodes are honest.

External observers and peer-to-peer nodes that are not mining nodes are considered malicious. Peer-to-peer nodes, if allowed, require no permission to operate. They are willing to spread disinformation and drop valid transactions if able.

5.3 The ACCORD Protocol

5.3.1 Mining Nodes

The mining network $\mathcal{N}$ consists of mining nodes $\{N_1, N_2, ..., N_i, ...N_{|\mathcal{N}|}\}$. Each mining node will have identifying information on the blockchain, including a set of public keys. These keys include a $\mathcal{K}_i$ standard key, which is a key on an elliptic curve, and an $\mathcal{A}_i$ additive signature key. These keys will be used for any signatures required. Since all nodes are aware of the other nodes' public keys, any node can produce a shared secret key with any other without the need for communication using the Diffie-Hellman Key Exchange. The states of a mining node are shown in Figure 5.2.

The credentials of all mining nodes are certified by the Membership Authority.

5.3.2 Membership Service Authority

Permissioned blockchains differ from permissionless blockchains in that only authorized parties are allowed to participate in the consensus protocol. This means that the network can control who is able to participate in the network. This type

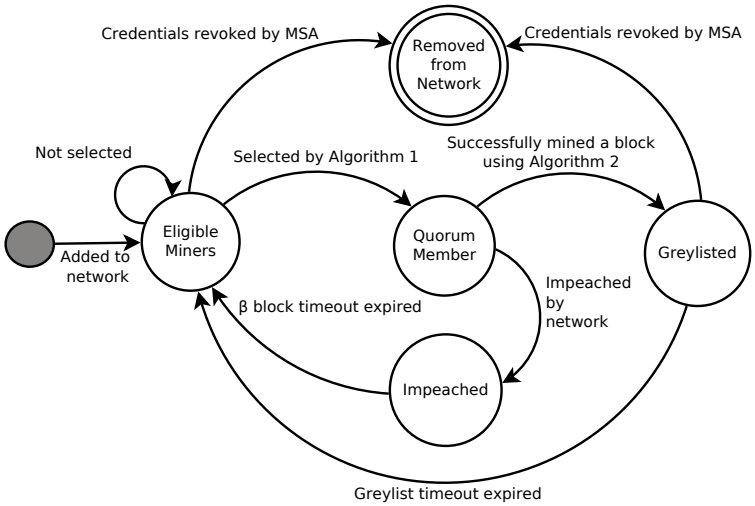


Figure 5.2: The paths a node travels through its various states.

of blockchain requires some degree of centralization to control its membership. This control is given to the *membership service authority* (MSA).

Before a party can join the network, they must contact the MSA to receive certification. The MSA is responsible for verifying a candidate's credentials and, if valid, certifying the candidate's public key. The validity of a candidate's credentials is determined by factors outside the blockchain (e.g., their status as a hospital).

The MSA acts as a certificate authority (CA) within the blockchain. The MSA certifies a node's public key by creating a transaction to be added to the blockchain. Once the transaction is added to the blockchain, the new node will be eligible to mine and perform actions on the blockchain after β blocks to ensure that the addition of the node does not allow manipulation of the miner selection process. β is defined in Section 5.3.5. Once the node's key is certified, they will use their appropriate key to create any signatures required by the mining process. If a node requires their key to be changed, the MSA can certify new keys by creating a transaction. If a node is caught acting maliciously or wishes to withdraw from the network, the MSA

can release a transaction decertifying the node's public key. With this approach, the network can easily come to a consensus as to the complete list of valid miners for each block simply by analyzing the blockchain. The MSA can put a block deadline n on a membership transaction entering the blockchain, such that a block n that does not include this transaction will be rejected by any honest node aware of the transaction.

Though the existence of the MSA requires a higher level of trust compared to a completely decentralized permissionless blockchain, this level of trust does not need to be absolute in a single party. There is no requirement that the MSA be a single party or external to the mining pool. The MSA could be a consortium of parties from within and outside of the network or even the entire mining pool. By distributing the role of MSA across multiple parties, it reduces the level of trust required for each party by increasing the number of parties that must be corrupted to corrupt the network. If the MSA is the mining pool, where an MSA transaction is valid if signed by a majority of the mining pool, then the trust of the MSA is reduced to the same level of trust that exists within the mining pool, which is that simple majority of the mining network behaves honestly.

Authentication protocols between MSA and nodes (i.e., determining valid credentials) are not investigated in this work, as it is beyond our scope of this paper.

5.3.3 Data Propagation

Our system relies uses a peer-to-peer network to transfer information across the network. This information includes transactions, blocks, and votes. All nodes within the mining pool participate in the network as peer-to-peer nodes. Optionally, other observers of the network may be allowed to participate in the peer-to-peer network but will not participate in the mining process. Any transactions or blocks received

by a node will be verified by that node before propagation.

A node that receives a valid transaction that has not appeared in a block (an unconfirmed transaction) will add this transaction to their *mempool*. If a node receives a block $\mathcal{B}_n$ but does not have a previous block $\mathcal{B}_{n-k}$, then they will request $\mathcal{B}_{n-k}$ from the sender of $\mathcal{B}_n$.

We assume that the peer-to-peer network operates correctly with up to 50% corruption of the mining network. Messages will always propagate across the network in less than target block time α, regardless of corruption or network failures up to 50%. However it is not guaranteed that every node will receive every message.

5.3.4 Quorum: A distributed-leader system

In a single leader consensus protocol, the leader needs to present a block to the rest of the nodes in the network. This approach has a few pitfalls which make it untenable for our security goals. Either the leader has control of the contents of the block, or the block creation process requires validation by a large number of nodes in the network. In the first case, the main issue is that this leader can effortlessly omit transactions and potentially release multiple blocks. In the second case, the protocol becomes inordinately expensive due to the amount of communication required as the network grows. Additionally, if our protocol is to avoid costly leader elections, a malicious leader may potentially gain the ability to influence the choice of future leaders. This could allow a subgroup of the network to take control of the mining process.

To mitigate this, ACCORD distributes the role of leader to a group of nodes called a *quorum*. Rather than having one leader decide which transactions to include or exclude, each of the quorum members provide a set of transactions they wish to

be added to the new block. The quorum members then perform a union operation on these sets of transactions to determine the content of a master block, $\mathcal{MB}$, which is to be the next block. A block is valid if more than a threshold δ of quorum members validate the block by signing it. The members of this quorum should only sign if all the transactions they expect are in the block. This reduces the ability of any individual node to omit transactions or manipulate the block. Algorithm 5 describes our quorum-based block creation protocol.

Definition 3 *Quorum.* A quorum is a group of q nodes that is currently allowed to build a block on behalf of the network.

The members of a quorum are selected using the quorum selection protocol, described in section 5.3.5.

Though our protocol does not require a specific quorum size q or threshold δ, we used the following formulas to define our value:

$$\delta = \max(\lceil 0.8 * q \rceil, \lfloor 0.9 * q \rfloor) \tag{5.1}$$

$$q = \lceil \log_2(|\mathcal{N}|) \rceil + 1 \tag{5.2}$$

If a quorum $\mathcal{Q}$ fails to produce a block within a specified time or performs a detectable malicious action, the other nodes will *impeach* them, voting them out and allowing a new quorum to take over. This process is automatic and discussed further in the Voting Rules section (section 5.4). If multiple quorums fail to mine the next block and get impeached, the threshold δ can be reduced to make the block building process easier. We decrease the threshold by two every third impeachment for that

block, with a minimum threshold of $\lceil q/2 \rceil + 1$. This allows us to achieve reasonable block times during major network outages, as seen in Section 5.5.2.

With the help of quorums, we can avoid high communication, and due to the small size of it, we can achieve higher throughput in the network.

5.3.5 Quorum member selection algorithm

In ACCORD, quorum members are selected by Algorithm 1. The purpose of this algorithm is to ensure that honest quorums are regularly selected in the presence of large malicious coalitions within the mining pool. We endeavor to do this without the need for a network-wide synchronization while also preventing too much foreknowledge of the quorum roster, as a malicious quorum member may attack their honest quorum counterparts to corrupt them or steal their keys given indefinite time.

This algorithm takes the list of all eligible miners and the headers of the βth and $(\beta + 1)$th previous blocks to determine the next quorum. The eligibility of miners is described in Section 5.3.6. We define β as the number of blocks in a cycle of miner selection. A larger β allows miners provides greater robustness against large-scale malicious action. The downside is that a larger β requires nodes to keep more blocks in active memory and reduces the responsiveness of alterations to the mining pool (adding new nodes or removing malicious ones, as discussed in Section 5.3.2, while also allowing more time for nodes to potentially corrupt their fellow quorum members. We recommend a β of 7, as it provides decent robustness and limits the time it takes to convince another node to go malicious or coordinate an attack on them to approximately 70 minutes. Algorithm 1 will be used to deterministically select the quorum through a selection function. Figure 5.3 illustrates the quorum selection process.

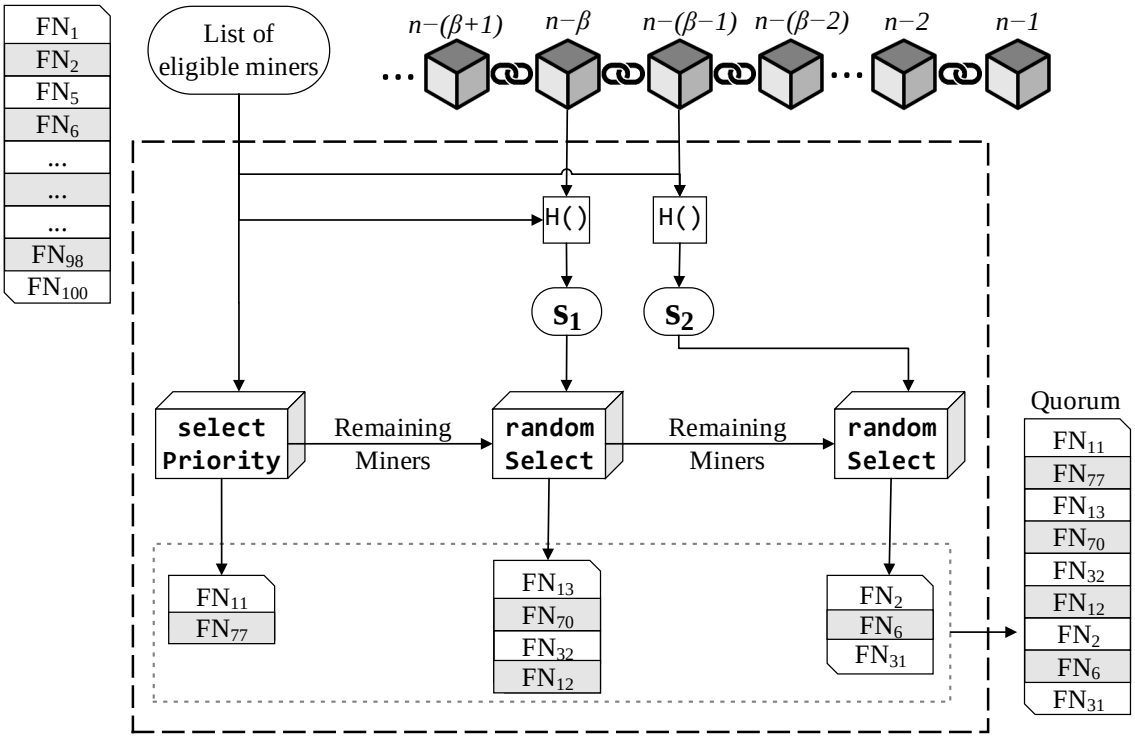


Figure 5.3: Selection process of quorum members (Algorithm 1).

First, any node that has been eligible to mine (e.g., not greylisted) for γ blocks are set aside to be selected as miners at the head of a quorum $\mathcal{Q}$. This priority list defined by γ ensures that everyone mines fairly regularly. γ is best defined as $a \cdot (\frac{n}{q})$. We use $a = 1.5$.

After these high-priority nodes are extracted, the remaining nodes for the quorum are selected. The header of the βth previous block is hashed with the list of eligible miners to create seed s_1. s_2 is similarly derived using the header of the $(\beta + 1)$th previous block. s_1 is then used to randomly choose half of the remaining miners needed for the quorum and adds them to $\mathcal{Q}$. Then, s_2 selects nodes from the mining pool and adds them to $\mathcal{Q}$ until $\mathcal{Q}$ is full. This process ensures that the choices made by the earlier block are not affected by the choices of the later block.

The network will be aware of the complete roster of $\mathcal{Q}$ when they become aware of the $(\beta+1)$th previous block. This process allows each node to determine the quorum roster independently, as there is no requirement to communicate with the network beyond propagation of the blocks. If $\mathcal{Q}$ is impeached, the quorum selection process is repeated using the two blocks previous to the seeds that chose $\mathcal{Q}$, the $(\beta-1)$th block and $(\beta-2)$th block, moving back two blocks every time there is an impeachment[5]. This new quorum will include the impeachment as part of their block as justification for mining.

While a round-robin approach is popular for leader selection in many protocols, we believe that with a large enough group, it can lead to some security issues. When a network consists of over 5000 nodes, specific future quorums can be known days in advance, which gives a malicious group enough time to potentially coerce or attack these future quorum members in an effort to compromise them. Nodes may even form coalitions based on their positions in the round-robin with miners near them. We believe that reducing the amount of time a quorum is known and randomizing its roster can be beneficial, as it increases the difficulty of such an attack.

5.3.6 Greylisting

Greylisting is a mechanism to prevent miners from being selected to mine in a quorum multiple times in quick succession. When a miner mines a block as a member of a quorum $\mathcal{Q}$, they will be added to the *greylist* and will not be eligible for selection by the Algorithm 1 until they are removed.

If primary quorum $\mathcal{Q}$ is impeached and an alternate quorum $\mathcal{Q}'$ produces the block, $\mathcal{Q}'$ will be added to the greylist and $\mathcal{Q}$ will not. However, the impeachment

[5]If this happens early in the history of the blockchain, the seeds can be cycled if headers before the genesis block are called for.

can not be allowed to alter how the next β quorums are selected. $\mathcal{Q}'$ addition to the greylist is deferred by β blocks as though they have not yet mined a block. These nodes can be selected to mine within the next β blocks and may even be selected as priority miners. $\mathcal{Q}$ will be ineligible to mine for β blocks, but they will not be greylisted. When this non-greylisted period of ineligibility ends, members of $\mathcal{Q}$ are returned to the list of eligible miners as though they were never greylisted. This means they may be immediately selected as a priority quorum member as though they were never made ineligible. The delay in the event of impeachment is to ensure that $\mathcal{Q}$'s impeachment has no impact on how quorums are selected outside of $\mathcal{Q}$'s defined influence.

In our analysis, we found that the greylist should consist of approximately 33% of the mining pool at any given time. Any excess nodes should be removed from the greylist if the list exceeds 33% of the nodes, following a FIFO structure. Greylisted nodes should still sign to accept blocks, propagate blocks, and potentially sign impeachment if warranted.

The main purpose of greylisting is to prevent a small malicious subset of the mining pool from being capable of taking control of the mining process. Since future quorums are selected by two blocks created by previous quorums, these two previous quorums have the ability to essentially select the future quorum. If the malicious party gets lucky enough to randomly gain control of β consecutive blocks, this could lead to them continually selecting their own members for mining duty. By having the greylist (and the priority miners), a malicious subgroup will be unable to select their members indefinitely. Another advantage of greylisting is to allow nodes to have a period of time where they do not need to maintain perfect liveness. For example,

after a quorum member mines and is greylisted, they can perform maintenance on their mining machine safely, knowing they will not be selected to a quorum in the near future.

5.3.7 Block structure

In our protocol, the block structure consists of a block header $\mathcal{H}$, metadata, transaction section, signature section, and the signatures from the quorum members $\sigma_{\mathcal{Q}}$. All included transactions will have their content and signatures separated. The content will be placed in the transaction section, and the signatures will be placed in the signature section. The block header is the SHA-256 hash of the metadata and transaction sections. The quorum members sign the hash of $\mathcal{H}$ concatenated with the hash of the signature section. Figure 5.4 illustrates the structure of a block.

Since $\mathcal{H}$ is used to select future quorum members, it is important that $\mathcal{H}$ be resistant to manipulation. To this end, we have created a system where there can only be one block header for a block containing a specific set of transactions mined by a specific quorum. This means that $\mathcal{H}$ is only dependent on the list of included transactions.

The metadata contains information about the structure of the block. This includes the number of transactions, the size of the sections, and the block header of the previous block. It also contains a list of identifiers for the members of the quorum (e.g., their public keys or identification numbers). If this block was created by a quorum selected due to impeachment, the metadata would also include a justification $\mathcal{J}$, which is essentially the impeachment of the previous quorum. The network signature associated with $\mathcal{J}$ will be included in the signature section. These signatures will be aggregated using the methods discussed in Section 5.3.8 to save space.

The transaction section includes two types of transactions: bookkeeping transactions and content transactions. The bookkeeping transactions are transactions related to the mining process, including MSA transactions to add and remove miners from the network. Included among these transactions are the quorum members' *null transactions*, discussed in Section 5.3.9. This group of transactions is expected to be small, possibly only consisting of the null transactions. They are sorted by transaction header at the beginning of the transaction section. The content transactions are the transactions containing the data that the blockchain is intended to distribute, (e.g., encrypted anonymous healthcare information). It will generally be the largest section, potentially containing thousands of transactions per block. The transactions in these sections are internally sorted by transaction header to ensure that the block header is solely dependent on the set of transactions included.

The signature section will include the signatures of the transactions in the same order or structure that they appear in the transaction section. The signatures are not included in $\mathcal{H}$, as one transaction could potentially have multiple valid signatures. To prevent manipulation of $\mathcal{H}$ in this manner, we do not include the signature section in $\mathcal{H}$.

In this blockchain, there is no defined size limit imposed on the blocks. Instead, quorum members attempt to add every valid transaction they are aware of to the block. This is a desired rule because every transaction should be added to the blockchain in a timely manner. This allows the number of transactions the network is able to process to be independent from the structure of the block or the consensus protocol.

The genesis blocks are the first β blocks. These blocks can be created by the MSA.

The first genesis block has no previous block header and contains the bookkeeping transactions to add the initial miners. The next $\beta - 1$ blocks are standard blocks except that they are created by the MSA, and they contain no transactions. These blocks form the beginning of the entire system from which the future blocks will be built.

A block is valid if it conforms to the rules of the blockchain for a block in its position. These rules include:

- All contained transactions are valid.
- The block structure is correct.
- The metadata is accurate.
- The block is signed by a valid quorum.

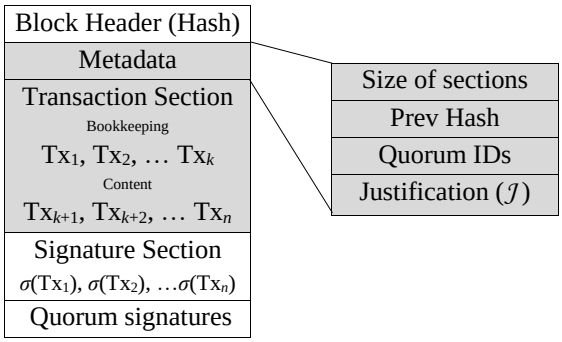


Figure 5.4: Block structure. The grey portions will be used in the Quorum Selection Protocol (Algorithm 1).

5.3.8 Additive Signature

Due to the number of signatures that need to be collected and maintained, it is important that these signatures be condensed. To this purpose, the voting signatures

will be aggregatable signatures to allow compression of the votes into one signature, accompanied with a bitmap to show who signed in the aggregate signature. Any arbitrary non-cooperative aggregate signature scheme will work, so we will employ the signature scheme created by Boneh et.al. [19]. To use this scheme (or many similar schemes), each signed message should be unique, so the messages can be appended with $\mathcal{K}_i$, their standard public key. Node N_i will sign these messages using $\mathcal{A}_i$.

5.3.9 Null Transaction

In our protocol, quorum members are selected using block headers from earlier in the blockchain. To ensure that each participating quorum member has an impact on the block header, they will be required to produce an empty null transaction for their block. Given a previous block header $\mathcal{H}_{n-1}$, there must be only one possible valid null transaction per node to prevent free manipulation of the block header. Additionally, this null transaction must be infeasible to forge without the node's private key. This is accomplished using a modified key image technique [78]. Let $\mathcal{K}_i$ be quorum member i's standard public key, s_i be their private key, and H_p be a hash-to-point function. Then, quorum member i's key image I_i is defined as follows:

$$I_i = s_i \cdot H_p(\mathcal{K}_i|\mathcal{H}_{n-1}) \tag{5.3}$$

Each party's null transaction will only contain I_i and proof that I_i is well-formed given the party's credentials. A Zero-Knowledge Proof of Diffie-Hellman pair will be used. The proof will act as the signature for this transaction and will have no impact on the block header $\mathcal{H}_n$, as it will be in the signature section of the block. A quorum member must provide a null transaction to sign the block.

5.3.10 Mempool

A node's mempool is where a node stores all valid unconfirmed transactions they have received. Upon receiving a transaction, the node verifies the transaction and adds it to the mempool. When a new valid block $\mathcal{B}$ is received by a node, all of the transactions in $\mathcal{B}$ are removed from that node's mempool. Each node maintains its own mempool. While a node is assigned to a quorum, it takes a snapshot of the mempool for use in the creation of the new block $\mathcal{B}$. While a node is a member of a quorum, they continue to receive transactions, but these transactions are not added to the block $\mathcal{B}$, as discussed in the Section 5.3.11.

5.3.11 Block Skeleton

A block skeleton is a list of transaction headers that a quorum has in their mempool and wishes to add to the block. A node's block skeletons act as a commitment to the list of transactions the node possesses. Upon receiving a block skeleton, fellow quorum members can use it to determine which transactions they need to request to build the completed block. The purpose of the block skeleton is to ensure that no malicious member of the quorum can create a transaction to modify the block header in a predictable way, which prevents the malicious node from influencing quorum selection. Block skeletons do not contain null transactions.

5.3.12 Communication in Block Creation

In our block creation protocol, each party attempts to create a direct secure line of communication between themselves and the other quorum members. Every party has a public key on the blockchain, so the execution of the Diffie-Hellman protocol is trivial and requires no communication between the nodes to create a shared key.

If a node cannot, for any reason, create a secure line of communication between themselves and another node, they can arrange for their messages to be forwarded by other quorum members or another node in the network. All messages from one quorum member to another should be signed by the sender, and the receiver should be responded to each message with a signed receipt.

If a quorum member P_i is attempting to send a message to another quorum member P_j and the direct communication fails (meaning they do not receive a receipt), they will broadcast their message to the rest of the quorum, asking them to forward the message. If they still do not receive a receipt, they will broadcast their message to the peer-to-peer network. The nodes on the peer-to-peer network will attempt to forward the message to P_j, who is then expected to broadcast a receipt to the peer-to-peer network, to be forwarded to P_i. The broadcast to the peer-to-peer network is to ensure that P_j cannot claim that P_i never sent a message that P_j chose not to respond to.

If a quorum member P_i is waiting for a specific message, they will send requests for the missing message to be forwarded to them to the other quorum members. After enough time has passed before, P_i will treat the respective parties P_j as absent or offline and attempt to continue with the protocol. While P_i is waiting, they will still respond to messages appropriately. This is also called `requesting`.

5.3.13 Block Creation Protocol

Before we begin the block creation protocol, Algorithm 1 is used to choose a quorum $\mathcal{Q} = \{P_1, P_2, ..., P_i, ..., P_q\}$.

The protocol starts with each P_i creating a block skeleton $\mathcal{SB}_i$ as a manifest of the transactions in their mempool. P_i creates a commitment $[\![\mathcal{SB}_i]\!]$ and sends $[\![\mathcal{SB}_i]\!]$

to the other nodes P_j in $\mathcal{Q}$. P_i now waits to receive $[\![\mathcal{SB}_j]\!]$ from all P_j. After this, P_i opens their commitment to each P_j, sending them $\mathcal{SB}_i$ and waiting to receive all $\mathcal{SB}_j$.

Once P_i has the skeleton blocks $\mathcal{SB}_j$ from all P_j, P_i will create a null transaction for this block and broadcast it to $\mathcal{Q}$. P_i then identifies the transactions from the $\mathcal{SB}_j$ that P_i does not possess. They then send a message requesting these transactions from P_j. If a requested transaction appears in multiple $\mathcal{SB}_j$'s, P_i will first request the transaction from the first node to their right in the quorum list[6], treating the quorum list as cyclical[7]. P_i now waits to receive all P_j's null transactions and all requested transactions.

Upon receiving all expected transactions, they are ready to construct the master block $\mathcal{MB}$. Each party should have the ability to construct $\mathcal{MB}_i$ locally following the block structure rules discussed in Section 5.3.7. When P_i locally creates a $\mathcal{MB}_i$, they need to ensure that all P_j generated identical $\mathcal{MB}_j$'s. This can be ensured by exchanging the block header $\mathcal{MH}_i$ with all other P_j. If there exist multiple distinct $\mathcal{MH}_j$, then the members of $\mathcal{Q}$ will exchange their blocks to find any discrepancies. If these discrepancies can be solved with forwarded messages, then the deviating P_j's can correct their blocks and produce an agreed-upon block header $\mathcal{H}_n$. If, after this, at least δ members of $\mathcal{Q}$ agree on a block, they will continue without the dissenting quorum members. If P_i agrees with $\mathcal{H}_n$, P_i will validate the block by signing their block as stated in Section 5.3.7. P_i should only sign at most one block to succeed $\mathcal{B}_{n-1}$ and refuse to sign any alternates that appear in the future. When the block header has more than δ signatures, P_i will add the quorum signatures $\sigma_{\mathcal{Q}}$ to the block $\mathcal{B}_n$ and

[6](e.g. if P_{i+1} and P_{i+2} both have the transaction, P_i will first ask P_{i+1})

[7]P_{q+j} is the same as P_j

broadcast $\mathcal{B}_n$ to the peer-to-peer network. $\mathcal{B}_n$ will be asynchronously validated by the network while the next quorum works to produce $\mathcal{B}_{n+1}$. $\mathcal{Q}_n$ sends $\mathcal{B}_n$ directly to $\mathcal{Q}_{n+1}$, $\mathcal{Q}'_{n+1}$, and $\mathcal{Q}'_n$, where $\mathcal{Q}'_x$ is the quorum responsible to mine if $\mathcal{Q}_x$ is impeached, so that they can start working as soon as possible. If $\mathcal{Q}_{n+1}$ or $\mathcal{Q}'_{n+1}$ has not received $\mathcal{B}_n$ within an expected time period, they will send a request to the peer-to-peer network asking for the block. $\mathcal{Q}_n$ should endeavour to release $\mathcal{B}_n$ approximately α time after receiving $\mathcal{B}_{n-1}$

Algorithm 1 Quorum members selection

To create a quorum to mine block n:
Input: list of x eligible miners $\mathcal{FN}_u = [FN_1, FN_2, ..., FN_k]$, quorum size q, priority threshold γ, and block headers $\mathcal{H}_{n-\beta}$ and $\mathcal{H}_{n-(\beta-1)}$

1: $s_1 \leftarrow h(\mathcal{H}_{n-\beta}|\mathcal{FN}_u)$
2: $s_2 \leftarrow h(\mathcal{H}_{n-(\beta-1)}|\mathcal{FN}_u)$
3: remove any impeached miners from $\mathcal{FN}_u$

- If $|\mathcal{FN}_u| < q$, reset $\mathcal{FN}_u$ to its original state.

4: $\mathcal{Q} \leftarrow \{\}$
5: **Select priority nodes:** Select all nodes $\mathcal{FN}_i$ from $\mathcal{FN}_u$ where the last instance of $\mathcal{FN}_i$ being removed from the greylist was before block $n - \gamma$. Add at most q of these nodes to quorum $\mathcal{Q}$.
6: s_1 **selects quorum members:** $\mathcal{Q}$ $\quad \mathcal{Q} \cup$ `randomSelect(`$\mathcal{FN}_u$`,` $\left\lceil \frac{q-|\mathcal{Q}|}{2} \right\rceil$`,` s_1`)`
7: s_2 **selects quorum members:** $\mathcal{Q}$ $\quad \mathcal{Q} \cup$ `randomSelect(`$\mathcal{FN}_u$`,` $q - |\mathcal{Q}|$`,` s_2`)`
8: return $\mathcal{Q}$

5.4 Mining rules

5.4.1 Block status definitions

A block $\mathcal{B}$ created by a quorum $\mathcal{Q}$ can exist in several different states in the view of an honest node.

Definition 4 *Pending.* $\mathcal{B}$ is *pending* if it has neither been *accepted* nor *abandoned*.

Definition 5 *Accepted:* $\mathcal{B}$ is *accepted* if at least $\left\lfloor \frac{|\mathcal{N}|}{2} \right\rfloor + 1$ of $\mathcal{N}$ have signed to accept $\mathcal{B}$ and all blocks previous to $\mathcal{B}$.

Algorithm 2 Building a Block $\mathcal{B}_n$

Input: A set of unconfirmed valid transactions (mempool) and a quorum $\mathcal{Q}$ of q nodes, $\{P_1, P_2, ...P_i, ...P_q\}$, with a required signing threshold of δ, and a time limit t_{limit}.
Output: A valid block or null if fails. If $\mathcal{Q}$ is *impeached*, abort the protocol

Overview: P_i is a member of $\mathcal{Q}$. This protocol takes the mempools of the nodes in $\mathcal{Q}$ and combines them to form a block.

Initialization. P_i creates a skeleton block $\mathcal{SB}_i$ with all available transaction headers from their mempool.
P_i creates a commitment $[\![\mathcal{SB}_i]\!] = f_{\text{comm}}(\mathcal{SB}_i, \kappa_i))$. // κ_i is a randomly generated key
`broadcast`$(P_i, \mathcal{Q}, [\![\mathcal{SB}_i]\!])$ // P_i exchanges commitments $[\![\mathcal{SB}_i]\!]$ with all P_j
`request`$(P_i, \mathcal{Q}, [\![\mathcal{SB}_j]\!])$
P_i creates their null transaction I_i.
`broadcast`$(P_i, \mathcal{Q}, (\mathcal{SB}_i, \kappa_i)$, `broadcast`$(P_i, \mathcal{Q}, I_i)$ // P_i opens commitment $[\![\mathcal{SB}_i]\!]$ for all P_j
`request`$(P_i, \mathcal{Q}, (\mathcal{SB}_j, \kappa_j))$, `request`$(P_i, \mathcal{Q}, I_j)$
P_i verifies $[\![\mathcal{SB}_j]\!]$ with $\mathcal{SB}_j$ and κ_j. If any P_j produces a bad $\mathcal{SB}_j$ or fail to produce or properly open their commitment $[\![\mathcal{SB}_j]\!]$, they are omitted from the mining process.
P_i verifies all I_j's using their included proofs. If I_j is invalid, P_j are omitted from the mining process.
P_i creates the skeleton master block $\mathcal{SMB}$ $\bigcup_{l \leftarrow 1}^{q} \mathcal{SB}_l$
for all transactions $\mathcal{T}_x \in \mathcal{SMB} - \mathcal{SB}_i$ **do**
    `request`$(P_i, \mathcal{Q}, \mathcal{T}_x)$ // All transactions in $\mathcal{SMB}$ that P_i does not know
end for
P_i creates the master block $\mathcal{MB}$ // P_i includes all transactions from $\mathcal{SMB}$ and all I_j
$\mathcal{H}'_n \leftarrow$ `block_hash`$(\mathcal{MB}_i)$ // P_i hashes the parts of the block used in the block header.
$\mathcal{MH}_i \leftarrow$ `full_block_hash`$(\mathcal{MB}_i)$ // P_i creates the hash they intend to sign.
`broadcast`$(P_i, \mathcal{Q}, \mathcal{MH}_i)$
`request`$(P_i, \mathcal{Q}, \mathcal{MH}_j)$
let z the size of the largest group in $\mathcal{Q}$ broadcasting the same $\mathcal{MH}$
if $z = q$ **then**
    $\mathcal{B}_n \leftarrow \mathcal{MB}_i$
    $\mathcal{H}_n \leftarrow \mathcal{H}'_n$
else if $z < q$ **then**
    `broadcast`$(P_i, \mathcal{Q}, \mathcal{MB}_i)$
    `request`$(P_i, \mathcal{Q}, \mathcal{MB}_j)$
    while $t_{\text{elapsed}} < t_{\text{limit}}$ or $z < \delta$ **do**
        Attempt to reconcile differences using message log to create $\mathcal{B}_n$. If there is evidence P_j performed malicious actions (e.g. creating multiple $\mathcal{SB}_j$), their malicious action is broadcasted to $\mathcal{Q}_n$ and their contributions to the block are omitted.
    end while
end if
if $\mathcal{MB}_i = \mathcal{B}_n$ **then**
    $\sigma_i =$ `block_sign`$_{P_i}(\mathcal{MB}_i)$
    `broadcast`$(P_i, \mathcal{Q}, \sigma_i)$
end if
`request`$(P_i, \mathcal{Q}, \sigma_j)$
let $z' \leftarrow$ the number of valid signatures of $\mathcal{MH}$
if $(z' \geq \delta)$ **then**
    Add all σ_j to $\mathcal{B}_n$
else
    return null
end if
return $\mathcal{B}_n$

Definition 6 *Abandoned:* $\mathcal{B}$ from $\mathcal{Q}$ is *abandoned* if:

1. *a competing block* $\mathcal{B}'$ *has become well-established or*

2. $\mathcal{Q}$ *has been impeached and* $\mathcal{B}$ *has not been well-established.*

Definition 7 *Well-established:* $\mathcal{B}_n$ is well-established if a chain of at least β' consecutive *accepted* blocks ($\mathcal{B}_{n+1}$, $\mathcal{B}_{n+2}$, ... $\mathcal{B}_{n+\beta'}$) are appended to it.[8]

When $\mathcal{B}_n$ is released by $\mathcal{Q}_n$, it is *pending* to all nodes that receive it. The quorum $\mathcal{Q}_{n+1}$ will operate assuming $\mathcal{B}_n$ will not be *abandoned*.

5.4.2 Voting Rules

In our system, there are multiple situations where a node N_i in $\mathcal{N}$ will be expected to vote. When N_i votes, they will release a unique standardized message and sign it using their additive signature key A_i. This message is dependent on what they are voting on. The two major causes of voting in our protocol are block accepting and impeachment. A similar approach can be used if $\mathcal{N}$ is the MSA to create MSA transactions.

Block acceptance procedure. When N_i receives a valid block $\mathcal{B}_n$, they wait for time α to ensure $\mathcal{B}_n$ propagates and there is no competing $\mathcal{B}'_n$, from the same quorum. If, after waiting α time, N_i has receives no conflicting blocks created by $\mathcal{Q}_n$ and $\mathcal{Q}_n$ has not been *impeached*, N_i will vote to accept $\mathcal{B}_n$. N_i votes by creating a message $M_A \leftarrow (\texttt{accept\_flag}|\mathcal{MH}_n|K_i)$, where `accept_flag` is a constant flag to identify block acceptance, $\mathcal{MH}_n$ is the full block header of $\mathcal{B}_n$, and K_i is N_i's standard key. M_A is then signed and broadcasted to the network.

[8]The number of blocks can be equivalent to the quorum selection block cycle β, meaning $\beta' = \beta$, but this is not a requirement.

N_i may sign a block $\mathcal{B}_n$ where previous blocks have not been accepted yet. However, N_i should not sign $\mathcal{B}_n$ if $\mathcal{B}_{n-\lfloor 0.5 \cdot \beta' \rfloor}$ has not been accepted, N_i should wait until $\mathcal{B}_{n-\lfloor 0.5 \cdot \beta' \rfloor}$ is accepted. This prevents a buildup of unconfirmed blocks while preventing the potential of a *well-established* fork.

Node N_i should not vote to accept multiple blocks from the same quorum. Any node N_j that votes for multiple blocks from the same quorum has committed a detectable malicious action and will be punished. However, N_i voting for acceptance of a block from $\mathcal{Q}_n$ does not prevent N_i voting for the impeachment of $\mathcal{Q}_n$.

If $\mathcal{Q}_n$ has been impeached, N_i should not sign any blocks appended to $\mathcal{Q}_n$'s block $\mathcal{B}_n$ unless $\mathcal{B}_n$ is *well-established*.

The purpose of waiting for time α (i.e., propagation) before voting is to make it improbable that an honest node N_i voted for $\mathcal{B}_n$ and another honest node N_j voted for a competing block $\mathcal{B}'_n$ from the same quorum.

Impeachment. *Impeachment* is the process by which $\mathcal{N}$ can remove malicious or defective quorums. There are three cases when N_i votes to impeach $\mathcal{Q}_n$.

The first case is if a node N_i has not received a valid block from quorum $\mathcal{Q}_n$ within $2 \cdot \alpha$ time of receiving $\mathcal{B}_{n-1}$. If $\mathcal{Q}_n$ was selected due to impeachment, N_i waits for $3 \cdot \alpha$ time after they signed the previous impeachment, assuming the previous impeachment was successful, before signing the next impeachment.

The second case is if N_i has received multiple valid blocks ($\mathcal{B}_n^1$, $\mathcal{B}_n^2$...) from $\mathcal{Q}_n$ with none of them being *accepted* within time $3 \cdot \alpha$, N_i will vote to *impeach* $\mathcal{Q}_n$. The quorum's malicious actions are clear to the network and would likely result in some form of punishment (e.g., expulsion from the network), even if they are not impeached.

The third case is if N_i has signed the block $\mathcal{B}_n$, but $\mathcal{B}_n$ has not been accepted for a period of $2 \cdot \alpha$. This could occur if there is a required MSA transaction with a block deadline of n that N_i is not aware, and this transaction is not present in the $\mathcal{B}_n$.

N_i will create a message $M_I = (\texttt{impeachment\_flag}|\mathcal{MH}_{n-1}|\mathcal{Q}_n|c|K_i)$, where `impeachment_flag` is a constant flag to identify impeachment, $\mathcal{MH}_{n-1}$ is the full header of the previous block, c is the number of previous impeachments on quorums creating a block on $\mathcal{MH}_{n-1}$, and K_i is N_i's standard public key. M_I is then signed and broadcasted to the network. Quorum members can vote for their own impeachment.

If $\mathcal{B}_n$ gets accepted and the impeachment on $\mathcal{Q}$ does not reach a majority, then node N_i should accept $\mathcal{B}_n$. If $\mathcal{B}_n$ gets accepted and the impeachment on $\mathcal{Q}$ succeeds, then node N_i should reject $\mathcal{B}_n$ unless $\mathcal{B}_n$ is *well-established*.

If N_i is part of $\mathcal{Q}'_n$, the quorum selected in the event of $\mathcal{Q}_n$'s impeachment, N_i can start to work with other members of $\mathcal{Q}'_n$ before $\mathcal{Q}_n$'s impeachment to decrease their response time in the event of impeachment.

It is important to note that impeachment is not necessarily the result of malicious action by $\mathcal{Q}_n$. For example, $\mathcal{Q}_n$ may fail if fewer than δ of them are online due to a network or power outage. Any impeachment, however, should result in an investigation outside the blockchain to determine whether any malicious action took place. Such an investigation can consult the message transcripts from our protocol.

5.4.3 Multiple Accepted Blocks

Due to the delay of α time before voting, it is highly unlikely that two honest nodes will sign to accept two competing blocks $\mathcal{B}_n$ and $\mathcal{B}'_n$ from the same $\mathcal{Q}_n$. However, it is theoretically possible for this to occur. If honest nodes divide their votes on a large scale, it is possible that both blocks may get a majority of the votes due to

malicious double voting (i.e., a node voting for multiple blocks). If N_i views multiple competing blocks as accepted, N_i will vote to impeach $\mathcal{Q}_n$, rejecting all $\mathcal{Q}_n$'s blocks unless one of them is *well-established*. For multiple blocks to become *well-established*, this attack would have to occur on β' consecutive blocks. The probability of this is highly improbable without a breakdown of the peer-to-peer network (e.g., a partition) or majority corruption of the mining pool. Additionally, double voting is an easily detectable malicious action and would result in punishment.

5.5 Experimental Evaluation

In this section, we evaluate the performance of our protocol in different simulation scenarios. These include analyzing communication benchmarks within the quorum, robustness against non-responsive miners, an analysis of an attempt to control the mining process, and miner selection distribution.

5.5.1 Communication costs

To determine the communication overhead of our consensus protocol, we will first measure the overhead within the quorum, then measure the communication overhead of the entire network. We will do this by measuring the messages required by the consensus that is not required for the basic functioning of the blockchain. For example, we measure the costs of transferring skeleton blocks and voting but not the cost of transferring transactions, as the transactions would have to be communicated in any consensus protocol.

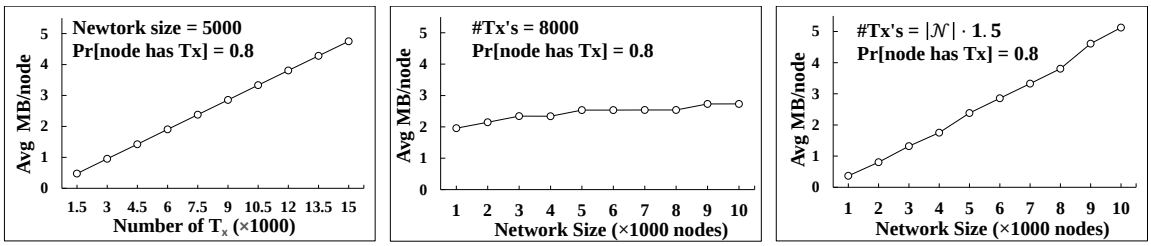
To measure the communication costs within the quorum, we conducted multiple groups of experiments with varying network sizes and the number of transactions. We measured the number of messages required for a quorum to create a block and

used this to calculate the total size of all the messages. We ran multiple experiments with different parameters. We measure the average size of the different types of messages sent with respect to transactions and network size. We assumed that a list of transactions exists in the network, and the probability that a given node has any given transaction is 80%. We ran these experiments with three methods of handling the variables. First, we changed the number of transactions while keeping the network constant at 5000 nodes. Second, we increased the size of the network while keeping the transactions constant at 8000. Finally, we had a more realistic scenario where the network size and number of transactions scale together such that the number of transactions is 1.5 times the size of the network.

We assume in this context that the nodes are behaving honestly and there are no network errors. The values are measured as the total communication of one node in the quorum.

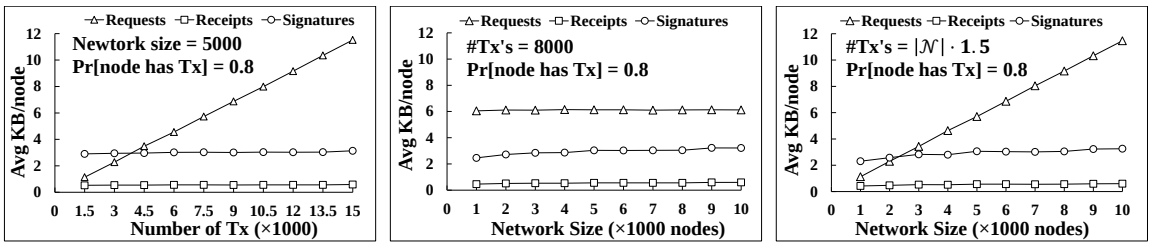
Figures 5.5 [a-c] shows the communication overhead of the exchange of skeleton blocks in MB. Figure 5.5 (a) shows that as the number of transactions increases, the size of the communications associated with the exchange of skeleton blocks scales linearly. In Figure 5.5 (b) we hold the number of transactions constant. By observing the graph, we can conclude that the average bandwidth used to exchange skeleton blocks appears to only increase when the size of the quorum is increased. In Figure 5.5 (c), the trends seen in Figure 5.5 (a) and Figure 5.5 (b) are apparent and there does not appear to be any emergent complexity.

Figures 5.6[a-c] show similar trends, but the scaling of these graphs shows these communications are negligible compared to the skeleton blocks.



(a) Scaling Tx (b) Scaling network size (c) Scaling network size and Tx

Figure 5.5: Average communication overhead per node from skeleton blocks with different number of transactions and nodes.



(a) Scaling Tx (b) Scaling network size (c) Scaling network size and Tx

Figure 5.6: Average communication overhead per node of smaller messages within quorum to build a Master block.

5.5.2 Fault Tolerance

In a distributed system, failures of nodes in the network are commonplace and a concern for the robustness of the system. A node failing to participate in the network could be due to a power or internet outage, an attack, or a simple malicious act by the node. These issues are not a theoretical concern (e.g., Gmail has power outages every couple of years [57]). Therefore we assume that any node can face an unexpected technical problem at any time. Furthermore, it is probable that if one node has a technical problem, many other nodes may be facing the same technical problem simultaneously. These fault tolerance experiments show the robustness of

our consensus protocol against different percentages of inactive nodes. We measure robustness by measuring average block times with different signing thresholds. We assume that the peer-to-peer network is still able to function with a maximum propagation time $\alpha = 10$ minutes. We assume that the nodes that are active in the network are behaving honestly in this context.

Figure 5.7 depicts the fault tolerance of different configurations of the protocol, namely the number of nodes initially required to sign a block. In this setup, if the attempts to mine the next block have three consecutive impeachments, then the number of quorum signatures required to mine this block is reduced by two without reducing the number of required signatures to below a simple majority.

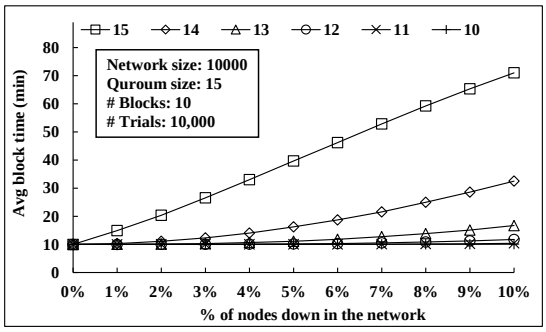
The x-axis indicates the total percentage of nodes in the network that are temporally unavailable (e.g., power outage, malicious behavior), and y-axis shows the average time in minutes required to form a functioning quorum. We ran our experiment with 10,000 iterations of a 10 block experiment. This is because in the event of a major outage or a node repeatedly not responding, and the MSA can temporarily remove the node from the mining pool and continue with the protocol. This should be achievable within generally be done within 10 blocks.

Figure 5.7(a) shows how the network responds to small outages of less than 10%. If we required every node in a quorum to sign a block, the average block time increases to untenable levels very quickly, taking almost an hour with a 10% network outage. The average time is significantly improved by requiring only 14 quorum signatures. This improved even further by only requiring 13 quorum signatures, reducing the average block time to less than 15 minutes to create a block with a 10% outage. While the block time is improved by reducing the signing threshold further, the effect

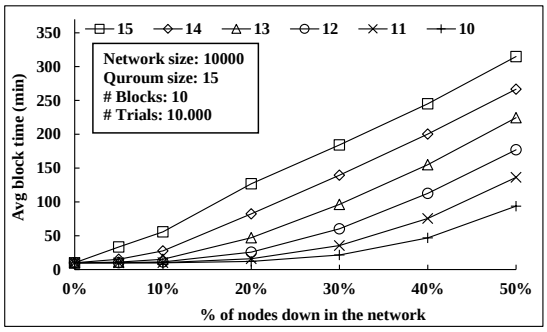
has apparent diminishing returns, and this reduction would make certain other attacks easier (see Section 5.5.3).

Figure 5.7(b) shows the recoverability of the system given larger outages of up to $50\% - 1$ of the network. Note that failure of more than $50\% - 1$ of the network would prevent the network from functioning. With any of the initial required signature settings, the block creation becomes intolerably slow for long-term use, but the system remains recoverable, with the 10 block run lasting no longer than 40 hours, at which point it can return to normal operation.

We observe from Figure 5.7 (a) and (b), the average block time is significantly increased, but the network still technically functional. Note that this block slowdown does not reduce the average throughput of transactions on the blockchain, as the size of blocks is unbounded, so a block that takes 60 minutes is, on average, 6 times larger than a block that takes 10 minutes.



(a) Blocktime up to 10% nodes unavailable



(b) Blocktime up to $(50\% - 1)$ nodes unavailable

Figure 5.7: Average block time given increasing network outages. $\alpha = 10$ **minutes.**

5.5.3 Manipulation in selection

A common concern with miner selection schemes that rely on aspects of the block to select future miners is that it potentially gives malicious miners the ability to select future miners, which could allow these malicious miners to take control of the system by repeatedly selecting members of their group to mine. With this experiment, we intend to show that our protocol is resistant to such an attack.

We conducted this experiment assuming a network size of 10,000 mining nodes and a quorum size of 15, and we chose $\beta = 7$. We assume that if the quorum is malicious, then they can select a block header that will produce a specific ordering of nodes of their choosing. We modeled this by replacing the standard selection process for headers created by a malicious quorum with a simple selection of nodes. We also assume that, at the beginning of the experiment, the malicious group spontaneously gained control of β consecutive blocks, enough to cycle their control of the system. We ran this experiment with all the valid threshold settings, similar to the experiment in subsection 5.5.2. Finally, we compared our two seed models with an equivalent single seed model. Our test involved selecting 100,000 quorums, and we ran this test 5 times and averaged the results.

The strategy employed by the different versions of this protocol are as follows: On the single seed version, if the $(n - \beta)$th quorum is malicious, they will attempt to make quorum n a malicious quorum with the minimum possible number of malicious nodes. They will prioritize malicious nodes that have most recently left the greylist (low priority nodes). They will then fill the rest of the quorum with honest nodes that have least recently been on the greylist (higher priority nodes). If there are too many honest nodes that have priority (at least $q - \delta$ nodes) or too few malicious

nodes are available, they will be unable to take control of quorum n and instead fill the quorum with only higher priority honest nodes to improve the probability that their malicious group will regain control in the future.

On the two seed versions, a single quorum has less control, needing control of the $n - \beta$ and $n - (\beta - 1)$ quorums to control quorum n. If they have control of both quorums, they will select nodes as in the single seed version. If they control the $(n - \beta)$ quorum but not the $n - (\beta - 1)$ quorum, they will fill their half of the selection with low priority malicious nodes to increase the probability that block n will be malicious as long as there are not too many honest priority nodes. If they control quorum $n - (\beta - 1)$ and not $n - \beta$, they will take control of quorum n if they are able. Their ability depends on whether the priority miners and the miners selected by quorum $n - \beta$ include fewer than $q - \delta$ honest nodes and If, in any of these cases, they do not have enough eligible nodes to produce a malicious quorum, they will fill the quorum with higher priority honest nodes.

It is important to note that while this strategy is not optimal, it to be a reasonable strategy to show that such an attack on our system is not trivial.

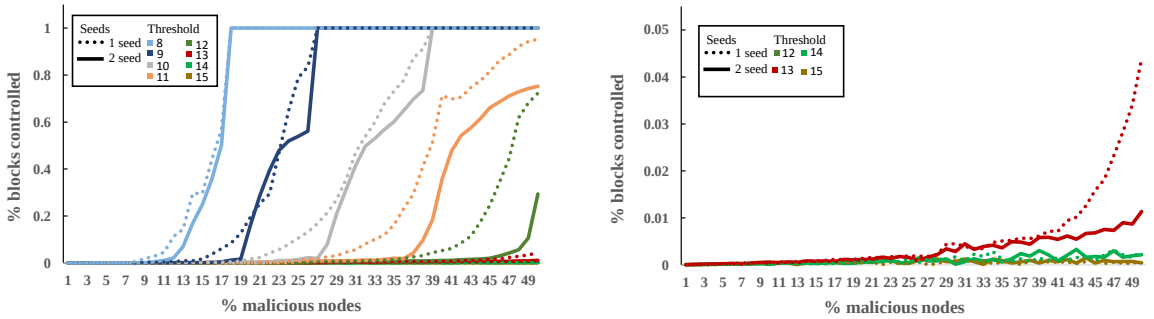
Figure 5.8(a) shows that requiring too low of a threshold makes the protocol too vulnerable to this attack, with the worst case of a threshold of 9 in the quorum causing the honest nodes to lose control at 15% malicious nodes. The results improve as the threshold increases, requiring that the malicious group use more of their membership per block and reducing their ability to clean the priority list of honest nodes. Figure 5.8(a) demonstrates that the single seed version generally allows the malicious group to have far greater control of the mining process up to a breaking point. This can be seen by the dotted lines representing the single seed version consistently being

significantly greater than the solid line of the same color, but both lines reaching 100% control at the same point.

On Figure 5.8(b), we remove the experimental runs where the percentage of controlled quorums ended up above 20% to get a better look at our recommended threshold settings as network corruption approaches 50%. This shows that the two seed versions of our recommended $\delta = \lfloor 0.9 * q \rfloor = 13$ threshold produces results roughly equivalent to the more secure thresholds, whereas its single seed counterpoints have approximately 4% of the blocks created being created by the malicious group. This, combined with our robustness experiments in Subsection 5.5.2, leads us to recommend this threshold.

We should note that when the malicious nodes were given control, they managed to keep control on our recommended settings for approximately 680 consecutive blocks at $50\% - 1$, or 4.7 days with a 10 minute block time. However, it is important to note that on none of the trials of the two-seeded version did the adversary manage to take control of the mining process on its own, even at $50\% - 1$. This is due to the fact that the two-seed process prevents a takeover of the quorums in chunks. In the single seed version, the malicious group is able to take control of one-βth of the network if they randomly gain control a single block for as long as they have members who can fill the positions and are not forced out by honest priority nodes. There taking control is as simple as randomly being selected β times. In the two seed version, a randomly selected malicious quorum results in this quorum having partial influence over two blocks instead of complete control over one, meaning they are unlikely to perpetuate their control unless they randomly get control of multiple consecutive blocks in a row. The probability of gaining control decreases with the two-seeded version as the period

increases.



(a) All thresholds **(b) Trimmed thresholds**

Figure 5.8: Percentage block corruption vs malicious control of mining pool. The solid line is our protocol, whereas the dotted line is an equivalent single seeded version, similar to existing work.

5.5.4 Miner Selection Distribution

To quantitatively analyze the models in terms of fairness, we designed and conducted several simulation experiments to observe the miner selection distribution. We endeavored to create a quorum selection protocol that, under honest conditions, will select every node to a quorum with approximately the same frequency. The assumptions we made for the experiments are as follows: All nodes are available at all times. Each participant is honest.

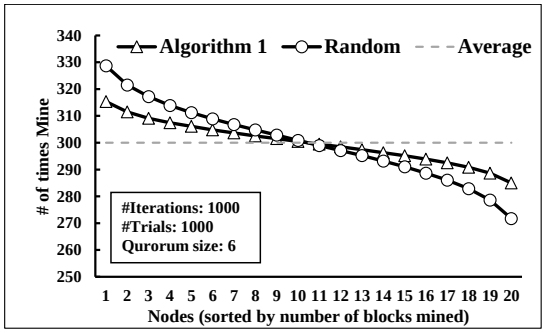
Figure 5.9(a) reported below is showing that the distribution of our protocol and a random selection. The x-axis represents the nodes based on their rank, meaning they are sorted by the number of times they mined. The y-axis indicates the number of times these nodes mined. We ran this experiment 1000 times, where each run simulated 1000 quorum selections. On each run, the nodes are ranked by the number of times they were selected to a quorum. The trials were averaged by rank to produce the graph.

Our quorum selection protocol produces more balanced results than random selection. This can be seen by observing that at every rank, our protocol assigned nodes to the quorum closer to the average than the random selection. At every point, our protocol produced results in values 50% closer to the average than the random selection.

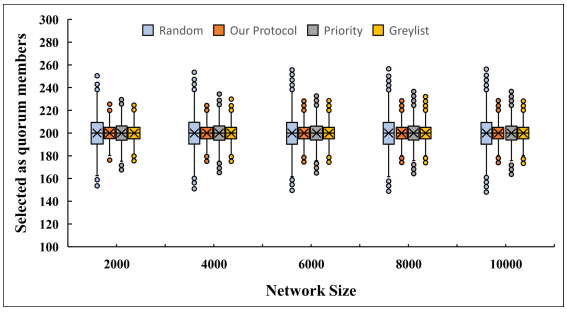
Figure 5.9(b) is a larger scale simulation of the quorum selection protocol. We simulated the protocol with several variations up to the point where the average number of times a node was selected was 200. We then ran this simulation 100 times and averaged the results by rank. We repeated this with larger network sizes. Figure 5.9(b) shows that the size of the network does not appear to affect the fairness of the selection process. It also shows that Algorithm 1 provides a much more even distribution on average when compared to random selection. Additionally, we analyzed the two major components of Algorithm 1 that impact selection distribution: the priority miners and the greylist. It appears that our fairness is mostly driven by the existence of the priority miners due to its enforcement of a pseudo-round-robin style of quorum selection. However, the greylisting does improve fairness as compared to random selection.

5.6 Threat-Risk Assessment Model

Consensus protocols for blockchains have certain common security vulnerabilities that need to be addressed. Below, we highlight a few of these vulnerabilities and our mitigations of these vulnerabilities.



(a) Frequency of quorum selection with 20 nodes.



(b) Distribution of quorum selection with increasing network size.

Figure 5.9: Distribution of quorum selection

5.6.1 Fork Resistance

Though a perfect immunity to forks is infeasible in a blockchain system, our protocol makes forks highly improbable in an honest setting, as well as making the network able to efficiently resolve forks in a malicious setting. We achieve this through four major procedures in our work: Quorum selection protocol , the block creation protocol, the block acceptance procedure, and the impeachment procedure.

Our quorum selection protocol (Algorithm 1) only selects one quorum to mine at any given time. This prevents multiple different quorums from creating blocks simultaneously and causing disagreement on the network. This protocol also helps to prevent forks by ensuring that if quorum $\mathcal{Q}_n$ created multiple blocks then the next quorum $\mathcal{Q}_{n+1}$ would be responsible for appending to each of these competing blocks. Our block creation protocol (Algorithm 5) requires δ signature from quorum members to produce blocks. It would require significant corruption of the quorum for them to be able to release multiple valid blocks.

The block acceptance procedure (Section 5.4.2) ensures that even if a quorum

releases multiple blocks, that it is highly improbable that different competing blocks will gain votes from honest nodes. Impeachment (Section 5.4.2) ensures that if a quorum releases multiple blocks, then network does not need to choose between these blocks (potentially resulting in a tie or no consensus) and can remove the malicious quorum.

5.6.2 Long-Range attack

In the Long-Range attack, an adversary gains enough control of the mining network at a certain block in the past. This allows them to write a new history of the blockchain, creating a fork from that point. Proof of Stake (PoS) is a consensus protocol that can suffer from this attack. In our system, this would theoretically be done if the adversary acquired enough private keys from miners who were members of the mining pool at that block. If the attacker accumulates the majority of the private keys, they may rerun the consensus protocol and write a new history of the blockchain from that point. As a countermeasure, our protocol has the concept of well-establishment, which is a rolling checkpoint system. From the perspective of a node or observer who has observed the blockchain between the block that is attacked and the time of attack, no one can reverse the blockchain before the last checkpoint. It should be noted that it is highly unlikely that the adversary could have enough transactions to carry their fork into the future as long as the transactions depend on other transactions' positions in the blockchain, meaning they would be unable to deceive participants who own transactions on the blockchain.

5.6.3 Future miners selection attack

In leader selection protocols where the leader's block is used to select future leaders, there is a concern that a malicious leader can manipulate their block to select a

new leader that they want rather than a random one. In the context of our protocol, this action would involve quorum members or the quorum as a whole to manipulate the block so that it would select future leaders favorable to them. We have several mitigations for this issue.

Our system uses blocks that are designed so that a set of transactions can only result in one block header $\mathcal{H}_n$. An honest quorum member P_i attempts to add all of the transactions from their *mempool* to the block $\mathcal{B}_n$ and expects all the valid transactions from the skeleton blocks created by the other quorum members to be added to $\mathcal{B}_n$ as well. Additionally, P_i expects null transaction from each participating member of $\mathcal{Q}_n$, not releasing their null transaction until all parties have committed to their skeleton blocks. Therefore, the ability of a minority of the $\mathcal{Q}_n$ to manipulate the $\mathcal{B}_n$ is limited, as the honest majority would not sign a block with omissions. This means that a block will not be manipulated by intentional omissions unless at least δ nodes in the $\mathcal{Q}_n$ are malicious.

If the adversary controls at least δ members of $\mathcal{Q}_n$, we assume they have full control of $\mathcal{Q}_n$. We assume they can perfectly select a set of transactions such that their block header selects their choice of nodes to future quorums within the scope of their header. This alone, however does not provide them with the ability to reliably create a malicious quorum, as each header selects half of two quorums $\mathcal{Q}_{n+\beta}$ and $\mathcal{Q}_{n+(\beta-1)}$ rather than one full quorum. To make $\mathcal{Q}_{n+\beta}$ malicious with certainty, $\mathcal{Q}_n$ and $\mathcal{Q}_{n+1}$ must both be malicious. Even if these two consecutive quorums are malicious, they will not be able to maintain control of the system (assuming $\beta > 2$). If $\mathcal{Q}_n$ and $\mathcal{Q}_{n+1}$ are both malicious, then they are able to control $\mathcal{Q}_{n+\beta}$, but no other quorums. This shows a diminishing of malicious control of one quorum every β blocks. Even if

the malicious group controls $\beta - 1$ consecutive quorums, their control of the system diminishes by one consecutive block every β blocks. To be able to maintain control, the malicious group would have to gain control of β consecutive quorums, which is highly improbable, even with $50\% - 1$ corruption of the network.

In the unlikely event that the adversary controls β consecutive quorums, the malicious group will have control over the system for a number of blocks. This time is limited, however, due to priority miners and the greylist. Even with $50\% - 1$ corruption with the malicious adversary using our malicious strategy on a network with our recommended settings, they were unable to maintain control of the network for more than a a couple days and never spontaneously gained control on their own. This is because they were unable to prevent honest nodes from obtaining priority. Additionally, to maintain control for any time, the malicious adversaries must be able to saturate the greylist with malicious nodes so that the malicious nodes entering the greylist are beign replace by malicious nodes leaving the greylist.

5.6.4 Stalling the Network

An alternative strategy for a malicious adversary is to create perpetual impeachments. The goal is to create a chain of non-functional quorums that are impeached and replaced with other non-functional quorums. This is done by ensuring each quorum has sufficient malicious membership to prevent the creation of a valid block. This is prevented by changing which quorums select the block when an impeachment occurs. To cause perpetual impeachment, the malicious group must have controlled consecutive quorums leading up to the current block with no network membership changes. Controlling these quorums implies a large number of malicious nodes would have to be in the greylist and would be ineligible to participate in such an attack. Ad-

ditionally, as the number of impeachments increases, the number of malicious nodes required to invalidate a block increases, therefore this attack would not be permanent.

5.7 Security Analysis

We will now provide a formal analysis to show that a minority in the quorum $\mathcal{Q}_n$ with q members cannot exercise significant control over the block header $\mathcal{H}_n$.

Proposition 1 *If a selected quorum $\mathcal{Q}_n$ with malicious membership $q' < \delta$ broadcasts a valid block $\mathcal{B}_n$ within their round n, then no member of $\mathcal{Q}_n$ can broadcast another valid block $\mathcal{B}'_n$ in round n such that $\mathcal{B}_n \neq \mathcal{B}'_n$.*

Proof. A valid block $\mathcal{B}_n$ must be signed by at least δ members of $\mathcal{Q}_n$. The honest $\mathcal{Q}_n$ members, who compose at least $q - (\delta - 1)$ members in the $\mathcal{Q}_n$, would be in communication and will, as a group, only sign one $\mathcal{B}_n$. Therefore, the remaining q' members would be incapable of creating a valid $\mathcal{B}'_n$ if $\mathcal{B}_n$ is valid, as $\mathcal{B}_n$ is signed by δ members of $\mathcal{Q}$.

To create a valid block $\mathcal{B}'_n$ without the participation of the honest members of $\mathcal{Q}_n$, the malicious members must either forge their signatures or find a block with an identical $\mathcal{MH}$. Both of these strategies are infeasible, as they depend on solving hard problems.

Proposition 2 *If a quorum $\mathcal{Q}_n$ with malicious membership $q' < \delta$ is selected and releases a valid block $\mathcal{B}_n$, then any valid T_x proposed by an honest member of $\mathcal{Q}_n$ will exist in $\mathcal{B}_n$.*

Proof. Assume P_i is an honest node and has transaction T_x in their mempool. P_i creates a skeleton block $\mathcal{SB}_i$, which includes T_x's transaction header. P_i sends

$\mathcal{SB}_i$ to all members of $\mathcal{Q}_n$, including the other honest nodes. The honest nodes will request T_x if they do not possess it. After verifying T_x, they will include T_x in their master block and produce a block $\mathcal{B}_n$. If the malicious nodes attempt to exclude T_x, they will be creating a different block $\mathcal{B}'_n$. The parties now compare their $\mathcal{MH}_i$. $P(\mathcal{MH}_n = \mathcal{MH}'_n | \mathcal{B}_n \neq \mathcal{B}'_n)$ is negligible $\left(\simeq \frac{1}{\sqrt{2^{129}}}\right)$, assuming a secure hash function with strong collision resistance. Since the honest nodes will not sign $\mathcal{MH}'_n$, the block that does not contain T_x will not be valid.

Proposition 3 *If a quorum $\mathcal{Q}_n$ with malicious membership $q' < \delta$ is selected, the ability of the malicious group to alter the block header $\mathcal{H}_n$ of a valid $\mathcal{B}_n$ is limited to $\sum_{k=q-\delta}^{q} \binom{q'}{q-k}$ possibilities.*

Proof. Assuming that a set of transactions can only result in one $\mathcal{H}_n$, the only mechanism to change $\mathcal{H}_n$ would be to change the set of transactions. Since each quorum member is expected to produce a null transaction and there exists one valid null transaction per quorum member, the contents of block $\mathcal{B}_n$ are unknown to the malicious parties before the beginning of the protocol. Therefore, the adversary does not have the ability to alter their entries in their block skeletons to produce predictable changes in the $\mathcal{H}_n$, as the null transactions are not released until the parties are committed to their skeleton block. The malicious parties, therefore, have only one point at which they can choose between different block headers. This point is when the honest nodes distribute their transactions. At this point, the adversary can know the full contents of $\mathcal{B}_n$ and can examine variants if combinations of their controlled parties are omitted. Essentially, the parties are able to choose either to continue with the protocol (and distribute their transactions) or stop participating.

Of the malicious group, up to $q - \delta$ can stop participating and $\mathcal{B}_n$ can still be valid. Therefore, the number of block headers they can choose from is $\sum_{k=q-\delta}^{q} \binom{q'}{q-k}$.

5.8 Conclusion

In this work, we have presented , a robust and efficient consensus protocol that can support the scale and critical nature of healthcare information. Our protocol achieves fairness with a miner selection protocol that does not require a network-wide synchronization and is capable of handling a large number of nodes. Finally, based on the extensive experimental evaluations, we determine it capable of functioning during major network outages and also capable of resisting attempts to manipulate the miner selection protocol. We also provide threat risk assessment model and security analysis of the proposed consensus protocol.

CHAPTER 6:

JANUS: TOWARD PREVENTING COUNTERFEITS IN PHARMACEUTICAL SUPPLY CHAINS UTILIZING A MULTI-QUORUM BLOCKCHAIN

6.1 Introduction

A supply chain is a network of linked stakeholders that process a product and pass it either up or down the chain [65]. In the pharmaceutical industry, prescription drugs are manufactured and ultimately passed down to an end user, typically a patient or a hospital. Major stakeholders in the pharmaceutical supply chain typically include suppliers, manufacturers, warehouses, distributors, pharmacies, and end users.

The Institute for Supply Management (ISM) conducted research in 2020 to survey global supply chains on the impact of COVID-19. Towards the end of March 2020, 95% of organizations in the survey reported that they already experienced disruptions as a result of the pandemic, or were expected to [3]. This shows how modern supply chains have been weakened. More specifically, the current pharmaceutical supply chain (PSC) suffers from a lack of traceability, security, and transparency. These faults ultimately contribute to the presence of illegitimate products in the market.

An illegitimate product can be any of the following: (1) a counterfeit product, (2) an adulterated product, (3) a product that is a part of a fraudulent transaction, or (4) a product otherwise deemed a hazard to users that is not fit to be dispensed [43]. Illegitimacy in the market in the form of counterfeits is arguably one of the most impactful to the PSC. The World Health Organization (WHO) estimates that the presence of counterfeit products in the pharmaceutical market can range anywhere from less than 1% in developed countries to over 10% in some developing countries [73]. These illegitimate products affect stakeholders throughout the chain. The industry suffers a net loss from production due to these faulty drugs entering the market. More urgently, counterfeit products can end up seriously harming or causing death to end users.

Beginning November 2023, the Food and Drug Administration (FDA) will require stakeholders (except for end users) in the PSC to comply with an act of guidelines called the Drug Supply Chain Security Act (DSCSA) [44]. The goal of the DSCSA is to create an electronic track-and-trace system for products in the PSC. In 2023, when the legislature is in full effect, stakeholders will be required to transmit all supply chain communications electronically and track their products at the individual package level [43]. Having an electronic system to track individual products throughout the PSC can significantly reduce the number of counterfeits in the market. The emerging applications of blockchain technology could be utilized to form an electronic, immutable, and decentralized system that provides traceability, security, and transparency through blockchain's inherent nature.

While blockchain can be used to form this immutable electronic system, the challenge of verification of physical products with digital data arises. It also poses the

challenge of ensuring end-to-end visibility. Current stakeholders in the PSC usually have their own local databases that store supply chain data which others in the PSC cannot access. By eliminating blind spots in the current system, end-to-end visibility can increase stakeholders' trust in the system and can also lead to any errors in the chain getting caught earlier on. Furthermore, blockchain maintains a ledger to provide sufficient tracking information that all involved stakeholders have access to, resulting in better coordination between the parties.

Researchers have already proposed blockchain-based solutions to modernize the PSC [6, 37]. Authors of [6] designed a system in which blocks are proposed when stakeholders in the chain initiate actions within the PSC (e.g., a warehouse sending out a shipment). For a block to be approved and added to their blockchain, a lead validator node must randomly select mining nodes from the network to validate it. Each package in the chain should have a near-field communication (NFC) tag, which holds product details, a counter, and a tag ID. Tags are read by receiving parties (e.g., a warehouse getting a shipment from a manufacturer) and checked to ensure that the product data and number of reads on the tag are correct. While NFC tags may be a beneficial aspect to connect the physical data of the PSC to its virtual blockchain, it is worth mentioning the security risks that they may pose. While NFC tags do require a relatively close scanning distance, they can still be easily scanned by almost anyone, and data stored on them can be read and potentially stolen. Furthermore, data on NFC tags can be overwritten. This could pose great threats to the PSC. Authors of [37] also propose the use of lead validator and regular validator nodes. However, in their system, the lead validator is responsible for the validation of transactions as well as the proposal of a block. Having a single leader in this position (that is solely

responsible for transaction validation) is a step towards centralization and requires more trust to rely on that node fully.

The objective of this paper is to design a trustless and scalable system for the pharmaceutical supply chain that: (1) achieves end-to-end visibility to prevent counterfeit drugs from entering the market and efficiently identify issues in the PSC process, (2) employs a decentralized decision-making protocol which eliminates the need for the stakeholders to trust each other while increasing their trust in the process, and (3) uses a quorum-based consensus protocol to ensure scalability. Our system, called JANUS [1], utilizes an cloning-resistant hologram tagging system that helps stakeholders trace products through the chain to confirm authenticity. It exploits the immutable nature of blockchain to increase transparency, security, and traceability while being fair, random, and scalable. It is targeted at the pharmaceutical industry because of these traits, as well as the fact that it complies with the DSCSA, which will be fully-enforced in November 2023 [44]. Our design ensures a tight link between the sequential steps in the physical supply chain process and the virtual blockchain, which is beneficial to complex markets such as the PSC.

6.1.1 Contributions

Our contributions in this paper are as follows:

1. We propose a novel blockchain-based pharmaceutical track-and-trace system, named JANUS, that prevents counterfeits from entering the pharmaceutical sup-

[1]We decided to name our system after the Roman myth of JANUS: God of beginnings and ends [89]. JANUS was portrayed having two faces: one facing forward and one facing back. We believe this to be an appropriate link to our research, as blockchain is an immutable ledger that allows users to build forward but also look back at previous blocks/transactions. Our system JANUS utilizes this backtracking ability to provide a means of tracing a product back to a stakeholder. This aids in determining authenticity of a product.

ply chain and ensures secure delivery between stakeholders.

2. Our design prevents any stakeholder in the system from introducing counterfeit products into the pharmaceutical market. We achieve this by utilizing nested hologram tags that identify where individual items are in the chain, providing end-to-end transparency of products in the system.

3. To maintain the security of the system, we introduce an equitable multi-quorum consensus protocol that achieves load-balancing among stakeholders of different types while maintaining fairness among stakeholders of the same type.

4. We implemented our system, including the multi-quorum consensus protocol. The results showed that JANUS is fair amongst stakeholders concerning mining contribution, and it is scalable with respect to a linear increase of nodes and transactions in the network.

6.1.2 Limitations

While end users, such as patients, are certainly stakeholders in the pharmaceutical supply chain, our model does not include them in the network. The network is designed to be private-permissioned, meaning only authorized nodes can join and view the blockchain and network activity. This means that end users would not have the ability to personally track their product's origin, limiting transparency at the patient level. Instead, our system relies on complete trust between the end user and their pharmacy.

While JANUS prevents stakeholders from acting dishonestly and claiming they never received a shipment when they actually did, there is always the potential for

honest mistakes, such as losing packages in delivery. This is a limitation of JANUS, as it does not have steps in place to detect or resolve these types of errors.

6.2 Proposed Solution

6.2.1 System Overview

JANUS provides a decentralized way to authenticate products in the pharmaceutical supply chain while preventing counterfeits from entering the market. To connect the physical aspects of the supply chain to the virtual data of the blockchain, JANUS utilizes hologram tags that hold critical information about the package that the tag occupies. This information can be used down the supply chain to identify a product and verify its legitimacy. After a physical inspection, the receiver is responsible for generating a transaction that notifies the network that the shipment has arrived. All transactions in the network are assigned to their appropriate quorums, as explained in section 6.2.4. These quorums are responsible for validating transactions and adding them to a proposed block. Once transactions have been validated, a separate quorum votes on the validity of the proposed block, thus determining whether or not the block should be added to the blockchain.

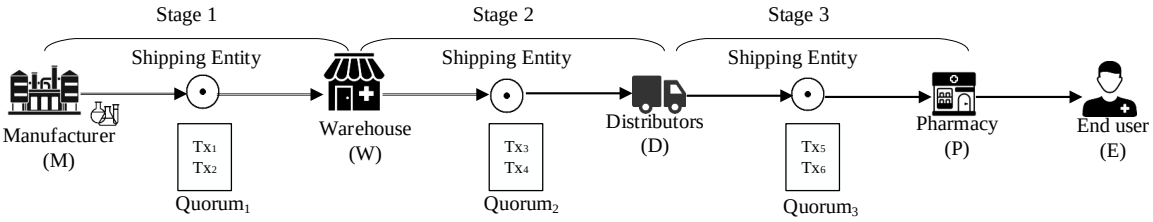


Figure 6.1: High-level overview of the system flow in which products move from stage to stage until they reach the end user.

6.2.2 Membership Service Authority

Our system uses a membership service authority (MSA) to ensure the integrity of the members in the network. The MSA is not a single entity in our design, thus not a single point of trust. We suggest that the MSA instead consist of all members in the network. The MSA certifies a potential network node's public key by creating a transaction that active members in the network can verify. Once the new entity is approved and has an eligible key, it can contribute to the network as an authorized member.

6.2.3 Notations

Table 6.1 contains key notations that will be used throughout the chapter.

Symbol	Definition
PSC	Pharmaceutical Supply Chain
M	Manufacturer
W	Warehouse
D	Distributor
P	Pharmacy
E	Shipping Entity
TID	Tag ID
PID	Product ID
S	Source Stakeholder
F	Destination Stakeholder
T_x	Transaction
t	Hologram Tag
p	Package
T'_x	Set of Proposed Transactions
$\mathcal{Q}$	Quorum
q	Number of Nodes
TH	Transaction Headers
B_Q	Block Quorum

Table 6.1: Table of notations.

6.2.4 Proposed Approach

Our proposed architecture establishes a trading and transmission mechanism to allow secure exchange between authorized entities in the PSC. The proposed model reflects a layered architecture that is categorized into two layers: physical and virtual.

The physical layer manages the cooperation between entities for physical products. These communications include the exchanging of goods along with proof of an auditable delivery (i.e., signed transactions). To track and trace the products, each package has a hologram tag. Tags are generated and placed by manufacturers onto each product. Once a product is created and ready to ship out, a tag is generated, holding the following information: tag ID (TID), product ID (PID), the product's National Drug Code (NDC), serial number, lot number, expiration date, and a list of descendent tags nested inside the tagged container. By having the descendant tag information in the parent tag, stakeholders can see which packages to expect inside a shipment. This nested system also allows stakeholders at any step in the chain to trace their product back to an authorized manufacturer, proving authenticity.

When a delivery arrives at its destination, the first phase is for the receiver to do a physical check on the shipment: (1) check the box for any obvious physical tampering, (2) check that the hologram tag has not been tampered with (i.e., the tag has been re-applied or ripped, indicating that the box has been opened or tampered with), and (3) scan the tag to ensure that the contents of its shipment are correct (check previous transaction's details, specifically PID and TID).

After receiving a shipment, the destination stakeholder initiates a transaction signifying that the shipment has been received. Transactions are aspects of the virtual layer, providing the essential connection between the physical data of the PSC and

the virtual data on the blockchain.

Algorithm 3 provides a step-by-step procedure of the processes that occur during each stage in the supply chain. In Figure 6.1, there are three primary stages: one from Manufacturer (M) to Warehouse (W), one from W to Distributor (D), and one from D to Pharmacy (P). In steps 1-11, the source stakeholder S fulfills an order made by the destination stakeholder $\mathcal{D}$. If S is a manufacturer, they are responsible for creating and placing all hologram tags that belong in the shipment and generating a transaction verifying that the order has been fulfilled. Otherwise, S just creates the transaction. Either way, the transaction gets broadcast to the whole network $\mathcal{N}$ for its appropriate quorum to validate. If the transaction is valid, it is added to a proposed block. Otherwise, it is rejected. In steps 12-19, S hands off the shipment to shipping entity E for delivery to $\mathcal{D}$. A copy of the first transaction made in step 4 is created and signed by E, signifying the successful pickup of the delivery. This signed transaction is broadcast to $\mathcal{N}$ and its appropriate quorum validates it. Just as in steps 8-11, it is added to a proposed block if deemed valid, and rejected if determined invalid. In steps 20-32, E makes the delivery to $\mathcal{D}$ and $\mathcal{D}$ must perform a physical check to ensure there has been no obvious tampering. If the inspection passes, $\mathcal{D}$ scans the hologram tag on the shipment, crosschecks the data, and generates a signed transaction $\sigma_{T_x,\mathcal{D}}$ notifying that the shipment has been received successfully by $\mathcal{D}$. This transaction is validated by its appropriate quorum and added to a proposed block if valid.

Transactions. The blockchain will be made up of different types of transactions, primarily:

Definition 8 *Source Transactions.* A transaction T_x is a source transaction if it is

Algorithm 3 Source-to-Destination Stage Delivery Process

Input: source stakeholder S, destination stakeholder $\mathcal{D}$, shipping entity E, transaction T_x, outermost hologram tag t_1, and the physical shipment/package itself p.
Output: $\sigma_{T_x,\mathcal{D}}$.

1: $\mathcal{D}$ places order to S for product
2: **if** S is a manufacturer:
3: S generates t for each product and package in the order:
t {tag ID, product ID, national drug code, serial number, lot number, expiration date, metadata}
4: S generates signed transaction $\sigma_{T_x,S} = Sig_s(T_x)$ indicating that the order from $\mathcal{D}$ has been fulfilled
5: $T_x \leftarrow$ {source ID, destination ID, product data, data of t_1}
6: $\sigma_{T_x,S}$ is broadcast to $\mathcal{N}$
7: Quorum validates $\sigma_{T_x,S}$
8: **if** $\sigma_{T_x,S}$ is invalid:
9: break
10: **else**:
11: $\sigma_{T_x,S}$ is added to proposed block and S proceeds to Step 12
12: S gives shipment to E for delivery to $\mathcal{D}$
13: E generates signed transaction $\sigma_{T_x,E} = Sig_E(\sigma_{T_x,S})$ to notify that p is in the delivery stage to $\mathcal{D}$
14: $\sigma_{T_x,E}$ is broadcast to $\mathcal{N}$
15: Quorum validates $\sigma_{T_x,E}$
16: **if** $\sigma_{T_x,E}$ is invalid:
17: break
18: **else**:
19: $\sigma_{T_x,E}$ is added to proposed block and E proceeds to Step 20
20: E arrives at the facility of $\mathcal{D}$ with p
21: $\mathcal{D}$ must perform a physical check to ensure that p has not been obviously tampered with
22: **if** p is noticeably tampered with:
23: break
24: **else**:
25: $\mathcal{D}$ continues to Step 26
26: $\mathcal{D}$ scans t_1 and generates signed transaction $\sigma_{T_x,\mathcal{D}} = Sig_{\mathcal{D}}(\sigma_{T_x,E})$ signifying that p has been received by $\mathcal{D}$
27: **return** $\sigma_{T_x,\mathcal{D}}$ to $\mathcal{N}$
28: Quorum validates $\sigma_{T_x,\mathcal{D}}$
29: **if** $\sigma_{T_x,\mathcal{D}}$ is invalid:
30: break
31: **else**:
32: $\sigma_{T_x,\mathcal{D}}$ is added to block and $\mathcal{D}$ proceeds to Step 33
33: **Repeat Step 1 through Step 33 until product has reached the pharmacy level**

generated and signed by a source stakeholder S in a stage ($\sigma_{T_x,S}$). □

Definition 9 *Shipping Transactions.* A transaction T_x is a shipping transaction signifying that a shipment has been sent out if it is a source transaction signed by a shipping entity E in a stage ($\sigma_{T_x,E}$). □

Definition 10 *Destination Transactions.* A transaction T_x is a destination transaction signifying that a delivery has been made to its destination if it is a shipping transaction signed by a destination stakeholder $\mathcal{D}$ in a stage ($\sigma_{T_x,\mathcal{D}}$). □

All transactions consist of the source (S), destination ($\mathcal{D}$), tag information of the outer tag (T_{i}), and the signature of the stakeholder initiating the transaction. If a transaction is generated but has no destination, the responsibility of validation will fall on the quorum of the highest order. For example, if W generates a transaction that is not about a product shipping out and therefore has no destination, validation will be done by Quorum 3, as defined in Section 6.2.4.

A transaction or proposed block must receive 2/3 valid votes from its quorum to be deemed valid.

Validation. Validation differs between transactions and blocks. To verify transactions generated as a product enters the chain, responsible quorums must check that S and $\mathcal{D}$ are both authorized addresses in the system, as well as check that the signature on the transaction comes from an authorized entity in the network. To verify transactions indicating that E is transporting a delivery, the S and $\mathcal{D}$ must match those on the previous transaction, and the signature of E must be an authorized member of the network. For transactions indicating an order has been delivered to its destination,

quorums must check that the new signature matches the destination of the original transaction and belongs to an authorized entity on the network. Members must also crosscheck the *TID* and the *PID* with the original transaction/order to ensure that the data matches.

Blocks are validated differently than transactions. Block quorum B_Q is responsible for computing the hash of the previous block in the chain and comparing it to the hash in the proposed block's header. If the hashes match and all quorum member signatures in the signature section are from authorized nodes, the block is considered valid.

Quorums. Our system takes advantage of the use of multiple quorums in order to achieve fairness, randomness, and scalability. In our system, there should be *N-1* quorum types, where N represents the number of stakeholder types in the network. Since we consider M, W, D, P, and E as primary stakeholders contributing to the network, four types of quorums would be formed. In reference to this model, we consider the following quorums:

Quorum 1: a quorum that consists of M, W, and E nodes

Quorum 2: a quorum that consists of W, D, and E nodes

Quorum 3: a quorum that consists of D, P, and E nodes

Quorum 4: a special block quorum B_Q that can consist of any stakeholder type in the network

Quorums are assigned to mine on their respective transactions. For example, Quorum 1 described above would be assigned to mine transactions that take place between M, W, and E. By having stakeholders mine on transactions of their own type, our system achieves local and global fairness. We define local fairness as the

fairness amongst stakeholders of the same type and global fairness as the fairness across the network amongst stakeholders of different types. For further explanation and implementation of local and global fairness, see Section 6.3.2.

As mentioned earlier, Quorum 4 is unique as it can consist of any stakeholder types in the network. This quorum, unlike others which validate transactions, is responsible solely for block validation.

All quorum members are selected randomly via our random-selection algorithm, which utilizes the hash of the previous block. This algorithm ensures local fairness between stakeholders in the same quorum pools. Entities that generate transactions will be responsible for running the random-selection algorithm. Because the algorithm relies on the hash of the previous block, all quorum members calculated will be the same even if multiple entities run it simultaneously.

Quorum member selection is outlined in Algorithm 4. To create quorum $\mathcal{Q}$, we first take the hash of the previous block. Our random-selection algorithm is performed using the number of nodes in the network n, the list of eligible miners in each quorum selection pool $\mathcal{G}$ where $\mathcal{G} = [g_1, g_2, ..., g_k]$, and the block header of the previous block in the chain $\mathcal{H}_r$. The algorithm randomly selects $\log(\mathcal{G})$ nodes to join a quorum.

Algorithm 4 Quorum Members Selection

Input: list of eligible miners $\mathcal{G}$, number of nodes in the network n, seed s_1, and block header $\mathcal{H}_r$ of the previous block

1: $\mathcal{Q} \leftarrow \{\}$
2: $s_1 \leftarrow h(\mathcal{H}_r)$
3: $\mathcal{Q} \leftarrow$ `randomSelect(`n`, `$\mathcal{G}$`, `s_1`)`
4: return $\mathcal{Q}$

Block Architecture. Each block in the chain will consist of a header, a body, and a signature section. The block's header will contain the hash of the previous block

as well as the timestamp of the current block's creation. The body will hold all of the valid transactions of the current block. Below the body, the signature section will hold the signatures of the quorum members that validated the transactions.

Block Validation. Block quorum B_Q must check that the block has proper structure, as well as compute the hash of the previous block and use it to run the selection algorithm. They can then check to make sure that all quorum members responsible for validating the transactions in the proposed block are authorized and participating honestly. In order for a block to be added to the blockchain, a minimum of 2/3 valid votes are required from B_Q.

6.2.5 Consensus Protocol

Blockchain requires a consensus protocol — a technique for establishing a single version of the records of transactions approved by the majority of participants. As our proposed design relies on a permissioned type blockchain where all nodes are known, a malicious participant would be discovered if it exercised to alter the chain in an unacceptable way. Therefore public consensus protocols such as Proof of Work (PoW) [69], Proof of Stake (PoS) [14], and Delegated Proof of Stake (DPoS) [14] are not perfect solutions. Some popular consensus protocols for private blockchain systems are Practical Byzantine Fault Tolerance (PBFT), Tendermint, and Hyperledger Fabric.

Many researchers and blockchain developers have started to focus on creating fair, scalable, and efficient consensus protocols that fit different use-cases. For example, [6] designed a protocol based on Tendermint to be used in the pharmaceutical supply chain. It relies on random selection for validator nodes, promoting fairness and some

degree of decentralization. While their protocol does have lead-validator nodes, they are responsible only for randomly-selecting $\log(n)$ validator nodes and broadcasting proposed blocks. Validators must go through two rounds of voting to reach consensus: pre-voting and pre-committing. At each round, responses from 2/3 of the $\log(n)$ validators must be received. After the pre-committing round, responses are counted and the final decision to reject or append the block to the blockchain is made.

In [37], a lead validator node was also proposed in the consensus protocol. However, in their design, the leader was responsible for proposing blocks as well as broadcasting them to the regular validators. These validators then check the validity of the block proposed by their leader, responding with a 0 to signify an invalid vote or a 1 to signify a valid vote. Whichever response receives more than half the votes determines if a block is added or rejected. Thus, for a lead validator to push a block to the chain, it must receive over 50% valid votes.

Our proposed consensus protocol relies on votes from authorized quorum members and does not use leader nodes. Instead, different quorums for each transaction type as well as a quorum designated for block validation vote to reach consensus on decisions regarding the blockchain. This provides a decentralized and trustless system.

Reaching Consensus. To reach consensus, our design requires all members of a quorum to place a vote of valid or invalid. The final decision is based on all responses received. A minimum of 2/3 valid votes is required for approval of any decision regarding the blockchain.

Algorithm 5 gives a step-by-step overview of how blocks are created and validated. In steps 1-3, quorum members view and share transactions that appear in their mempools. Quorum member Q_i requests the transactions from all other members' mem-

pools, with all other members being Q_j. By requesting each other's transactions, they can assure everyone has the same view. In steps 4-5, quorum members create a dummy block including all valid transactions from mempools of all members. Transactions missing in the dummy block are also then requested from others, as per steps 6-7. Once all transactions have been received, Q_i follows steps 8-10 and builds a full drafted block and hashes it to create a signed hash of the drafted block. It is then broadcast to the quorum and requests the signed hashes of drafted blocks created by the remaining members in the network to complete steps 11-12. If the hashed draft block achieves the threshold of 2/3 valid votes, members append their signatures and forward it to the block quorum responsible for validation. If the block achieves at least 2/3 of signatures from the block quorum, it is added to the main chain as per step 19.

6.3 Experimental Evaluation

6.3.1 Setup and Environment

All of our experiments were performed using a Windows 64-bit machine running Windows 10 Pro. The machine has an Intel i7-4810MQ CPU and 16GB of RAM. Our tests were written in C++ and compiled and executed in Windows Visual Studio. Because C++ cannot natively accommodate numbers as large as the standard 256-bit hash, we used the first 1/4 of the previous block hash to compute the quorum members. The full source code can be accessed here [2].

The following subsections describe and observe the results of the experiments we performed to test the fairness and scalability of our system, as well as the likelihood of

[2]https://drive.google.com/drive/folders/1IozLEXLQ5esLj-bIzo7ZCAOw5XfEnxnJ?usp=sharing

Algorithm 5 Building and Mining a Block $\mathcal{B}_n$

Input: A set of transactions T'_x with authorized signatures that have not yet been added to the chain, a quorum $\mathcal{Q}$ of q nodes, and a threshold of δ where δ is 2/3, and a time limit t_{limit}.
Output: If successful, a valid block is generated. Otherwise, it will return null.

1: **Initialization.** Members of $\mathcal{Q}$ take all headers of transactions TH that exist in their mempool.
2: `broadcast`$(Q_i, \mathcal{Q}, [\![\mathcal{TH}_i]\!])$
3: `request`$(Q_i, \mathcal{Q}, [\![\mathcal{TH}_j]\!])$
4: Q_i verifies $[\![\mathcal{TH}_j]\!]$. If any Q_j produces a falsified $\mathcal{TH}_j$, that transaction is rejected.
5: Q_i creates a dummy block $\mathcal{DB}$ that contains all valid transactions from their mempool and all Q_j mempools
$\mathcal{DB} \leftarrow \bigcup_{i \leftarrow 1}^{q} \mathcal{TH}_{ij}$
6: **for all** transactions $T'_x \in \mathcal{DB} - \mathcal{TH}_i$ **do**
7: `request`$(Q_i, \mathcal{Q}, T'_x)$
8: **end for**
9: Q_i builds a drafted block $\mathcal{DRB}$
10: $\mathcal{H}_n \leftarrow$ `block_hash`$(\mathcal{DRB}_i)$
11: Q_i generates $\mathcal{H}_{n_{Q_i}}$ by appending its signature onto $\mathcal{H}_n$
12: `broadcast`$(Q_i, \mathcal{Q}, \mathcal{H}_{n_{Q_i}})$
13: `request`$(Q_i, \mathcal{Q}, \mathcal{H}_{n_{Q_j}})$
14: **if** $\mathcal{H}_n$ achieves δ:
All members of $\mathcal{Q}$ append their signatures onto DRB and forward it to the block quorum $\mathcal{BQ}$.
15: **else**:
16: Repeat Step 2 through Step 12.
if $t_{\text{elapsed}} < t_{\text{limit}}$
17: break
18: `broadcast`$(\mathcal{Q}, \mathcal{BQ}, \mathcal{H}_{n_{Q_i}})$
19: **if** $\mathcal{H}_{n_{Q_i}}$ achieves majority of the signatures from $\mathcal{BQ}$:
It will be added to the main chain
20: **else**:
Null

a malicious quorum forming. We have also provided the results of the communication cost evaluation of our system.

6.3.2 Fairness

In this experiment, we assess the fairness of our algorithm on two scales: local and global. Local fairness refers to the balance of work amongst stakeholders of the same type, while global fairness refers to the load-balancing achieved across the system as a whole.

To test the fairness of our system, we assume there are four primary stakeholder

types, each represented with 20 nodes on the x-axis, while the y-axis represents the number of times a node was selected. Three quorums are generated in each iteration. We ran the test 5,000 times to simulate the generation of 5,000 blocks in the network, for a total of 15,000 quorums.

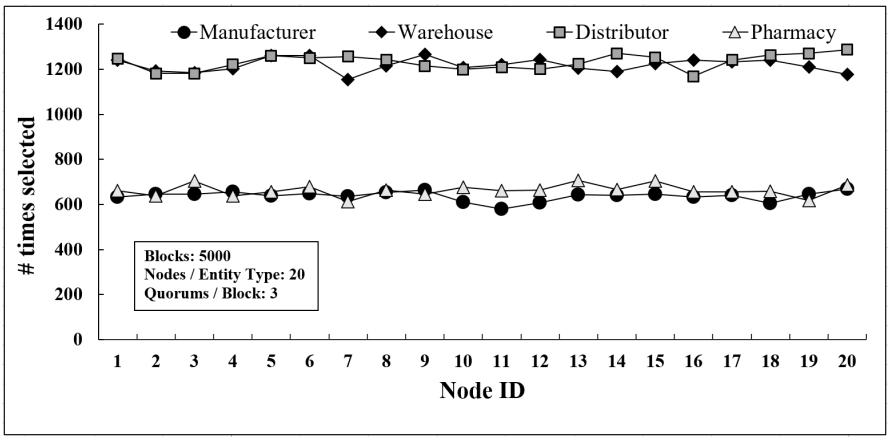


Figure 6.2: Local and global fairness.

Figure 6.2 illustrates the results of the fairness experiment. We observe that each stakeholder has a linear projection regardless of their type, where nodes were selected roughly the same number of times throughout the 5,000 trials. This indicates that we achieve local fairness. Looking at the graph as a whole, we can also see that our system achieves global fairness because certain stakeholders are selected for quorums more frequently than others. Warehouses and distributors are involved in double the transactions, justifying why they are selected roughly 1,200 times versus 600 like manufacturers and pharmacies. Thus, our algorithm accomplishes the task of being globally fair amongst stakeholders across the network.

6.3.3 Scalability

Scalability is crucial for an efficient system, especially one the size of the pharmaceutical supply chain. To assess the scalability of our design, we consider a linear

increase in the number of nodes and transactions in the network.

To test the scalability of JANUS, we track the runtime in seconds (y-axis) that it takes for quorums in networks with different numbers of nodes (x-axis) to synchronize their transaction information when there are a large number of transactions. We ran this test 100 times, each iteration recording responses for networks with 1,000 to 5,000 nodes sharing 2,000 to 8,000 transactions.

Figure 6.3 graphs the results of our scalability experiment. We observe a slight increase in runtime as the number of nodes progresses from 1,000 to 3,000 for all numbers of transactions. After our system reached 3,000 nodes, it began to level off. This is because quorum size for the subsequent networks is the same. Our graph shows a gradual increase in runtime that is consistently proportional between network size (number of nodes) and number of transactions. Thus, our system is proven to be scalable.

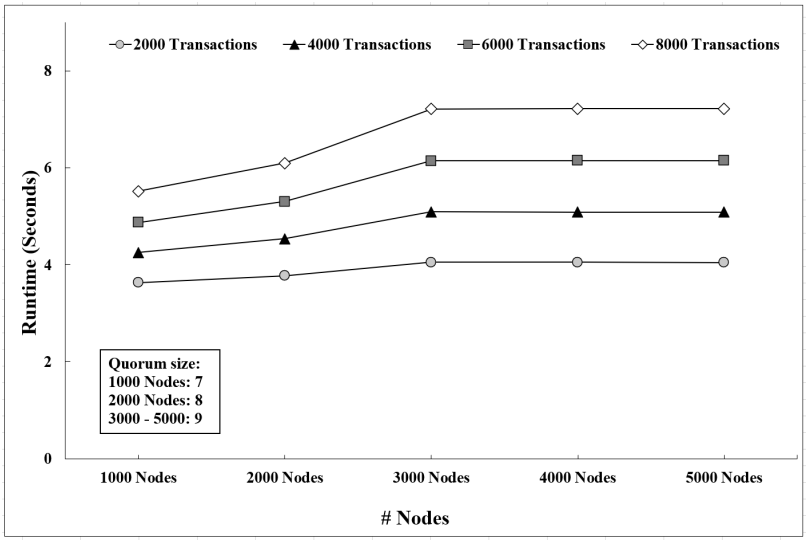


Figure 6.3: Scalability as nodes/transactions increase.

6.3.4 Resiliency Against Malicious Quorums

To prove the security of our algorithm, we performed an experiment to test resiliency against malicious quorums. To assess the resiliency, we consider that a percentage of nodes in the network are malicious to determine the frequency in which malicious quorums are formed.

To test resiliency against malicious quorums, we consider an increasing percentage of malicious nodes in the network (x-axis) and examine how it affects the total percentage of malicious quorums formed (y-axis). This experiment was repeated 100 times, considering 10%, 15%, 20%, 25%, and 30% malicious nodes in the network.

Figure 6.4 visualizes the results of our experiment. We observe that as the number of malicious nodes in the network increases, the percentage of malicious quorums increases exponentially. Based on this observation, it's fair to assume that the larger the network, the less probable a malicious quorum is to form (in comparison to smaller networks). When around 23% of nodes in a network are malicious, we define a threshold where the percentage of malicious quorums forming ranges from 0% to 1%. After this threshold, the percentage begins to increase exponentially. We want our network to remain 23% malicious or less to ensure no more than 1% malicious quorums are formed, which is a reasonable expectation in a permissioned network like ours.

This graph represents the case in which a regular quorum is malicious, but the block quorum is honest (or vice versa). It is worth mentioning that the probability of both types of quorums being malicious is exponentially lower.

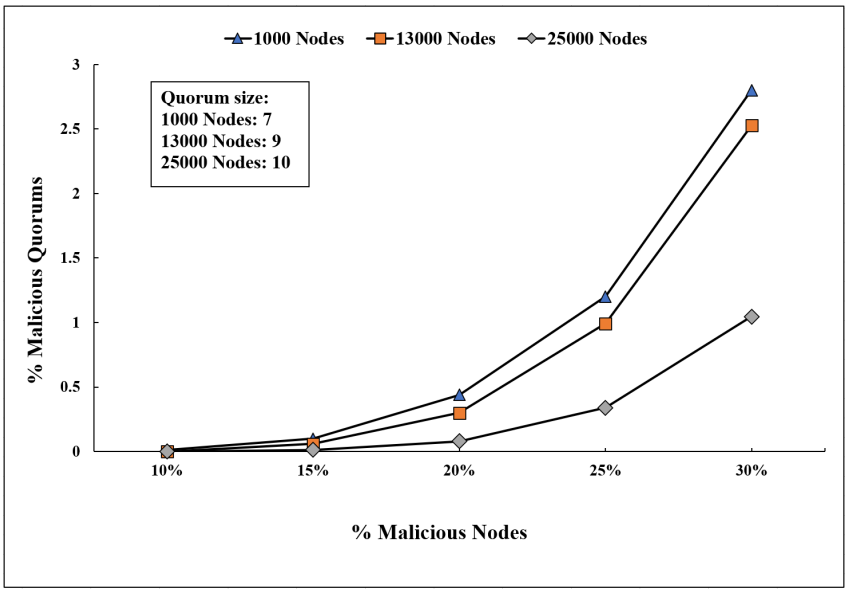


Figure 6.4: Potential percentage of malicious quorums forming.

6.3.5 Communication Cost

To evaluate the communication cost of JANUS, we first examined the cost of transactions in the network. Figure 6.5 visualizes the transactional cost in terms of average megabytes (MB) per quorum (y-axis) depending on the total number of transactions (x-axis). We ran this evaluation simulating networks of 2000, 4000, and 8000 nodes transmitting 2000 to 16000 transactions. We chose these network sizes in order to establish the transaction costs at different common quorum sizes. To build a draft block that is the same for all quorum members, we assume that the probability of any given node having any given transaction is at least 65%.

We observe that more nodes in a network results in longer processing times. As anticipated, it takes longer to process the same number of transactions in larger networks, as they require communication with more nodes.

Regardless of network size, communication cost will increase linearly as the num-

ber of transactions increases. This contributes to the scalability of JANUS, showing that it can handle as many nodes as possible without drastically affecting communication cost.

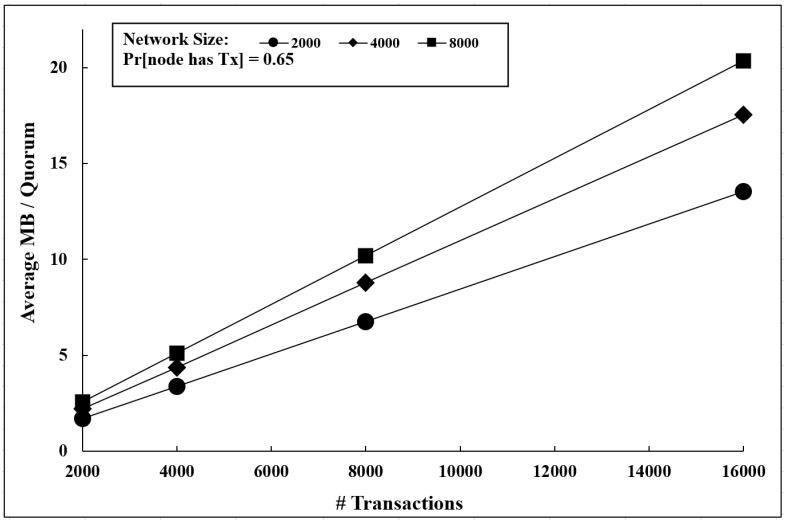


Figure 6.5: Communication cost as nodes/transactions increase.

6.4 Threats, Attacks, and Security Model

Evaluating the security of consensus protocols is challenging due to the variation of attacks encountered by blockchain systems. Threat modeling is a simple study directed by most researchers to systematically approach cyber threats and recognize potential system security concerns in advance.

We identify two threats: (1) quorum misbehaviour: a timing fault due to a miner transmitting self-contradictory blocks at the same time, and (2) denial of service: an omission weakness due to quorum members bypassing signing or announcing a transaction.

We find it important to also mention two key attacks that quorum models may be vulnerable to: eclipse attacks and random manipulation attacks. Eclipse attacks

can devastate a system by allowing an adversarial quorum member to attack other quorum members. Here, the malicious quorum member can monopolize all of the victim's incoming and outgoing connections, hence separating the victim from the rest of its quorum members. In this way, the adversary can modify the victim's view of the draft block. The malicious node could target multiple quorum members simultaneously.

Randomness manipulation attacks occur when a malicious quorum makes attempts to permutate the order of all transactions until it is confirmed that a malicious quorum will be formed in the future. The probability can be dramatically decreased if we choose the seed as a concatenation of the hash of the previous block and hash of the previous to previous block in Algorithm 4.

Proposition 4 *If a selected quorum has malicious nodes $< \delta$ in a round, then all malicious nodes will add all the valid T_x to its block in that round.*

Proof. Assume the majority of the members are honest, then honest members will receive the valid T_x from others. Due to the majority, any invalid T_x forwarded by the malicious node will be discarded.

Let's consider the following situation: A dishonest member does not accept a block within a pre-defined waiting period, but all honest members send their votes to the draft block. As long as δ is satisfied, then all honest peers will make the same update of their blockchain.

Proposition 5 *Assume one of the proposing quorum Q_n is faulty in a round. If majority of other quorums remain honest, then it is impossible for them to add invalid blocks to their blockchain in the same round.*

Proof. In JANUS, a draft block is appended to the main chain if and only if the block quorum accepts it. As far as majority of this quorum remain honest, no invalid blocks can be added in case of other becomes malicious.

6.5 Conclusion

We have proposed a pharmaceutical-specific blockchain system that utilizes cloning-resistant hologram tags to aid in the prevention of counterfeit products from entering the pharmaceutical market. We evaluated JANUS against three metrics: fairness, scalability, and resiliency. Based on our implementation and large scale evaluation of the system, we have shown that our design maintains approximately similar workloads between all stakeholders, is scalable for large networks such as the pharmaceutical supply chain, and is resilient against malicious quorums. We conclude that blockchain technology has the potential to make the supply chain management system more transparent, traceable, and resilient.

CHAPTER 7:

CONCLUSION

In this dissertation, we performed extensive research on distributed consensus protocols and proposed three different protocols especially for private, healthcare, and supply chain domains. By the experimental evaluation of our approaches, we observe that fairness and effectiveness are the two exceptional properties for designing secure consensus protocols in blockchain applications.

In chapter 2, we elaborate related work that has been done over the years in the field of our dissertation research. We discuss the advantages and disadvantages of existing consensus protocols in the public, private, healthcare, and supply chain domain. We also present a comparative evaluation of the existing protocols related to cryptocurrencies.

In chapter 3, we presented and discussed the background knowledge needed for this dissertation work. We explain TEE, SGX, the abstract model of PoEt, remote attestation, and DSCSA policy.

In chapter 4, we propose our solution for building a consensus protocol that achieves higher throughput and is free from the collision. We evaluate our protocol against three metrics: throughput, scalability, and fairness. Finally, based on the simulation and large-scale evaluation, we conclude that our protocol outperforms intel's PoET in terms of throughput.

In chapter 5, we propose a novel permissioned multi-leader (quorum-based) consensus protocol that achieves fork-resistance, robustness, and scalability that can support the sheer scale and critical nature of healthcare information. Subsequently, based on the comprehensive evaluations we discovered it could remain functional during significant network interruptions and capable of resisting attempts to manipulate the miner selection protocol. We also provide a threat risk assessment model and security analysis of the proposed consensus framework.

In chapter 6, we propose a pharmaceutical-specific blockchain system that uses cloning-resistant hologram tags to prevent counterfeit products from entering the pharmaceutical market. Our implementation shows that our design keeps relatively comparable workloads between all stakeholders, is scalable for comprehensive networks, and is resilient against malicious quorums.

In a nutshell, we propose in this dissertation three different consensus protocols, and evaluate their overall performance based on metrics such as throughput, scalability, fairness, as shown in Table 1.1.

Looking Ahead. Designing and deploying a consensus protocol is a challenging task as it requires the analysis of various essential issues such as resiliency against node failures, malicious behavior of nodes, fork resistance, and transaction ordering. It would be interesting to explore how to determine the unique features of a consensus protocol that would be a good fit for all sectors and be used as a mainstream consensus protocol. As future work, we plan to modify the node-side protocol of PoQ by adjusting the quantum time depending on the time left of executing nodes to optimize the overall fairness of the protocol. Also, ACCORD could theoretically be extended by investigating how to make the selections of the quorum selection protocol only

known to the quorum until block creation to prevent DDOS attacks. In JANUS, it would be interesting to explore how to utilize blockchain to securely handle returns at any stage from a destination to a source throughout the pharmaceutical supply chain.

BIBLIOGRAPHY

[1] Why EHR data interoperability is such a mess in 3 charts, Oct 2018. [Online; accessed 9. Feb. 2021].

[2] Change Healthcare: enterprise blockchain with 30 million transactions per day - Ledger Insights - enterprise blockchain, Feb 2019. [Online; accessed 28. Jan. 2021].

[3] Can blockchain rescue from supply chain disruptions due to covid-19?, Dec 2020. Available at `https://www.devdiscourse.com/article/technology/1043874-can-blockchain-rescue-from-supply-chain-disruptions-due-to-covid-19`.

[4] Medical records 10x more valuable to hackers than credit card information: Cybersecurity issues in healthcare have fallen to criticism lately in light of rising data breaches, notably the hacked server at Franklin, Tenn.-based Community Health Systems which compromised protected health information of 4.5 million patients., Nov 2020. [Online; accessed 17. Nov. 2020].

[5] Mansoor Ahmed and Kari Kostiainen. Identity aging: Efficient blockchain consensus. 2018.

[6] Naif Alzahrani and Nirupama Bulusu. Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain. In *Proceedings of the*

1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, page 30–35. Association for Computing Machinery, 2018.

[7] Sébastien Andreina, Jens-Matthias Bohli, Ghassan O. Karame, Wenting Li, and Giorgia Azzurra Marson. Pots - a secure proof of tee-stake for permissionless blockchains. *IACR Cryptology ePrint Archive*, 2018:1135, 2018.

[8] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[9] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE, 2016.

[10] LM Bach, Branko Mihaljevic, and Mario Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. IEEE, 2018.

[11] Arati Baliga. Understanding blockchain consensus models. In *Persistent*. 2017.

[12] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok : Consensus in the age of blockchains. 2017.

[13] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok: Consensus in the age of blockchains. 2017.

[14] G. Bashar, G. Hill, S. Singha, P. Marella, G. G. Dagher, and J. Xiao. Contextualizing consensus protocols in blockchain: A short survey. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 190–195, 2019.

[15] Golam Dastoger Bashar, Alejandro Anzola Avila, and Gaby G Dagher. Poq: A consensus protocol for private blockchains using intel sgx. In *International Conference on Security and Privacy in Communication Systems*, pages 141–160. Springer, 2020.

[16] B. M. A. L. Basnayake and C. Rajapakse. A blockchain-based decentralized system to ensure the transparency of organic food supply chain. In *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pages 103–107, 2019.

[17] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[18] Arpan Bhattacharjee, Shahriar Badsha, Abdur R Shahid, Hanif Livani, and Shamik Sengupta. Block-phasor: A decentralized blockchain framework to enhance security of synchrophasor. In *2020 IEEE Kansas Power and Energy Conference (KPEC)*, pages 1–6. IEEE, 2020.

[19] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 416–432. Springer, 2003.

[20] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Research perspectives and challenges for bitcoin and cryptocurrencies (e.v.). *IACR*, 2015.

[21] Ethan Buchman. Tendermint: Byzantine fault tolerance in the age of blockchains. Master's thesis, The University of Guelph, 2016.

[22] Vitalik Buterin et al. Ethereum: A next-generation smart contract and decentralized application platform. 7, 2014.

[23] Kelly Caine, Spencer Kohn, Carrie Lawrence, Rima Hanania, Eric M Meslin, and William M Tierney. Designing a patient-centered user interface for access decisions about ehr data: implications from patient interviews. *Journal of general internal medicine*, 30(1):7–16, 2015.

[24] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[25] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In *SSS*, 2017.

[26] Xusheng Chen and Shixiong Zhao. Scalable, efficient, and consistent consensus for blockchains, 2018.

[27] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[28] Amie Corso. Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework. 2019.

[29] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016(086):1–118, 2016.

[30] Gaby G Dagher, Jordan Mohler, Matea Milojkovic, and Praneeth Babu Marella. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable cities and society*, 39:283–297, 2018.

[31] Hung Dang, Anh Dinh, Ee-Chien Chang, and Beng Chin Ooi. Chain of trust: Can trusted hardware help scaling blockchains? *ArXiv*, abs/1804.00399, 2018.

[32] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 123–140, New York, NY, USA, 2019. ACM.

[33] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. 2018.

[34] Vikram Dhillon, David Metcalf, and Max Hooper. The hyperledger project. In *Blockchain enabled applications*, pages 139–149. Springer, 2017.

[35] Omar Dib, Kei-Leo Brousmiche, Antoine Durand, Eric Thea, and Elyes Ben Hamida. Consortium blockchains: Overview, applications and challenges. *International Journal On Advances in Telecommunications*, 11(1&2), 2018.

[36] Alevtina Dubovitskaya, Zhigang Xu, Samuel Ryu, Michael Schumacher, and Fusheng Wang. Secure and trustable electronic medical records sharing using blockchain. In *AMIA annual symposium proceedings*, volume 2017, page 650. American Medical Informatics Association, 2017.

[37] Sanjeev Kumar Dwivedi, Ruhul Amin, and Satyanarayana Vollala. Blockchain based secured information sharing protocol in supply chain management system with key distribution mechanism. *Journal of Information Security and Applications*, 54:102554, 2020.

[38] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology*, pages 139–147, 1993.

[39] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer, 2015.

[40] Kai Fan, Shangyang Wang, Yanhui Ren, Hui Li, and Yintang Yang. Medblock: Efficient and secure medical data sharing via blockchain. *Journal of medical systems*, 42(8):136, 2018.

[41] Nilgun Fescioglu-Unver, Sung Hee Choi, Dongmok Sheen, and Soundar Kumara. Rfid in production and service systems: Technology, applications and issues. *Information Systems Frontiers*, 17, 01 2014.

[42] Michael J. Fischer, Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. pages 1–7, 1983.

[43] Food and Drug Administration. Title II of the Drug Quality and Security Act, 2014. Available at https://www.fda.gov/drugs/drug-supply-chain-security-act-dscsa/title-ii-drug-quality-and-security-act.

[44] Food and Drug Administration. FDA In Brief: FDA provides new guidance to further enhance the security of prescription drugs in the U.S. supply chain, Jun 2021. Available at https://www.fda.gov/news-events/press-announcements/fda-brief-fda-provides-new-guidance-further-enhance-security-prescription-drug

[45] F Greer, C McLean, and TE Graham. Caffeine, performance, and metabolism during repeated wingate exercise tests. *Journal of applied physiology*, 85(4):1502–1508, 1998.

[46] Ijazul Haq and Olivier Muselemu. Blockchain Technology in Pharmaceutical Industry to Prevent Counterfeit Drugs. *International Journal of Computer Applications*, 180:8–12, 03 2018.

[47] Anton Hasselgren, Katina Kralevska, Danilo Gligoroski, Sindre A Pedersen, and Arild Faxvaag. Blockchain in healthcare and health sciences—a scoping review. *International Journal of Medical Informatics*, 134:104040, 2020.

[48] Of Health and U. S. Department Human Services. HHS Finalizes Historic Rules to Provide Patients More Control of Their Health Data. *HHS*, Mar 2020.

[49] J. P. Howard and M. E. Vachino. Blockchain compliance with federal cryptographic information-processing standards. *IEEE Security Privacy*, 18(1):65–70, 2020.

[50] Jun Huang, Xiang Li, Cong-Cong Xing, Wei Wang, Kun Hua, and Song Guo. DTD: A Novel Double-Track Approach to Clone Detection for RFID-Enabled Supply Chains. *IEEE Transactions on Emerging Topics in Computing*, 5(1):134–140, 2017.

[51] Drew Ivan. Moving toward a blockchain-based method for the secure storage of patient records. In *ONC/NIST Use of Blockchain for Healthcare and Research Workshop. Gaithersburg, Maryland, United States: ONC/NIST*, pages 1–11, 2016.

[52] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols (extended abstract). pages 258–272, 1999.

[53] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. Blochie: a blockchain-based platform for healthcare information exchange. In *2018 ieee international conference on smart computing (smartcomp)*, pages 49–56. IEEE, 2018.

[54] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel® software guard extensions: Epid provisioning and attestation services. *White Paper*, 1:1–10, 2016.

[55] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. *IACR Cryptol. ePrint Arch.*, 2020:269, 2020.

[56] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[57] Alexis Kleinman. Gmail And Google Drive Are Experiencing Issues, And Naturally People Are Complaining About It On Twitter (UPDATE: It's Fixed). *HuffPost*, Dec 2017.

[58] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[59] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[60] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[61] Fátima Leal, Adriana E. Chis, Simon Caton, Horacio González–Vélez, Juan M. García–Gómez, Marta Durá, Angel Sánchez–García, Carlos Sáez, Anthony Karageorgos, Vassilis C. Gerogiannis, Apostolos Xenakis, Efthymios Lallas, Theodoros Ntounas, Eleni Vasileiou, Georgios Mountzouris, Barbara Otti, Penelope Pucci, Rossano Papini, David Cerrai, and Mariola Mier. Smart Pharmaceutical Manufacturing: Ensuring End-to-End Traceability and Data Integrity in Medicine Production. *Big Data Research*, 24:100172, 2021.

[62] Ling Li. Technology designed to combat fakes in the global supply chain. *Business Horizons*, 56(2):167–177, 2013.

[63] Xiaoguang Liu, Ziqing Wang, Chunhua Jin, Fagen Li, and Gaoping Li. A blockchain-based medical data sharing and protection scheme. *IEEE Access*, 7:118943–118953, 2019.

[64] Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. *IEEE Transactions on Dependable and Secure Computing*, 3(3):202–215, July 2006.

[65] John T. Mentzer, William DeWitt, James S. Keebler, Soonhong Min, Nancy W. Nix, Carlo D. Smith, and Zach G. Zacharia. Defining Supply Chain Management. *Journal of Business logistics*, 22(2):1–25, 2001.

[66] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. Proof of luck: An efficient blockchain consensus protocol. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*, SysTEX '16. ACM, 2016.

[67] Saikat Mondal, Kanishka P. Wijewardena, Saranraj Karuppuswami, Nitya Kriti, Deepak Kumar, and Premjeet Chahal. Blockchain Inspired RFID-Based Information Architecture for Food Supply Chain. *IEEE Internet of Things Journal*, 6(3):5803–5813, 2019.

[68] Takuro Nakagawa and Naohiro Hayashibara. Energy efficient raft consensus algorithm. In *International Conference on Network-Based Information Systems*, pages 719–727. Springer, 2017.

[69] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[70] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at `https://bitcoin.org/bitcoin.pdf`.

[71] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[72] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 305–319, 2014.

[73] World Health Organization. Counterfeit Medicines: an update on estimates 15 November 2006, 2006. Available at `https://www.who.int/medicines/services/counterfeit/impact/TheNewEstimatesCounterfeit.pdf`.

[74] Vishal Patel. A framework for secure and decentralized sharing of medical imaging data via blockchain consensus. *Health informatics journal*, 25(4):1398–1411, 2019.

[75] Yang Peng, Tomoyuki Nagase, Toshiki Kanamoto, Tsutomu Zeniya, and Shan You. A Virtual Optical Holographic Encryption System Using Expanded Diffie-Hellman Algorithm. *IEEE Access*, 9:22071–22077, 2021.

[76] Kevin Peterson, Rammohan Deeduvanu, Pradip Kanjamala, and Kelly Boles. A blockchain-based approach to health information exchange networks. In *Proc. NIST Workshop Blockchain Healthcare*, volume 1, pages 1–10, 2016.

[77] Lauren C Ramsay, Sarah A Buchan, Robert G Stirling, Benjamin J Cowling, Shuo Feng, Jeffrey C Kwong, and Bryna F Warshawsky. Retracted article: The

impact of repeated vaccination on influenza vaccine effectiveness: a systematic review and meta-analysis. *BMC medicine*, 15(1):1–18, 2017.

[78] Nicolas van Saberhagen. Monero white paper. Oct 2013.

[79] Fahad Saleh. Blockchain without waste: Proof-of-stake. 2018.

[80] M. Schöner, Dimitris Kourouklis, P. Sandner, E. Gonzalez, and Jonas Förster. Blockchain Technology in the Pharmaceutical Industry. 2017.

[81] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8), 2014.

[82] Affaf Shahid, Ahmad Almogren, Nadeem Javaid, Fahad Ahmad Al-Zahrani, Mansour Zuair, and Masoom Alam. Blockchain-Based Agri-Food Supply Chain: A Complete Solution. *IEEE Access*, 8:69230–69243, 2020.

[83] Michail Sidorov, Ming Tze Ong, Ravivarma Vikneswaren Sridharan, Junya Nakamura, Ren Ohmura, and Jing Huey Khor. Ultralightweight Mutual Authentication RFID Protocol for Blockchain Enabled Supply Chains. *IEEE Access*, 7:7273–7285, 2019.

[84] Harish Sukhwani, José M Martínez, Xiaolin Chang, Kishor S Trivedi, and Andy Rindos. Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pages 253–255. IEEE, 2017.

[85] Fei Tang, Shuai Ma, Yong Xiang, and Changlu Lin. An efficient authentication scheme for blockchain-based electronic health records. *IEEE access*, 7:41678–41689, 2019.

[86] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer, 2015.

[87] Abdul Wahab and Waqas Mehmood. Survey of consensus protocols, 2018.

[88] Jo Waller, Kirsten McCaffery, Henry Kitchener, James Nazroo, and Jane Wardle. Women's experiences of repeated hpv testing in the context of cervical cancer screening: a qualitative study. *Psycho-Oncology: Journal of the Psychological, Social and Behavioral Dimensions of Cancer*, 16(3):196–204, 2007.

[89] Donald L. Wasson. Janus, Feb 2015. Available at `https://www.worldhistory.org/Janus/`.

[90] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE, 2018.

[91] Qinghan Xiao, Thomas Gibbons, and Hervé; Lebrun. RFID Technology, Security Vulnerabilities, and Countermeasures. In Yanfang Huo and Fu Jia, editors, *Supply Chain*, chapter 19. IntechOpen, Rijeka, 2009.

[92] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks, 2019.

[93] Manaf Zghaibeh, Umer Farooq, Najam Ul Hasan, and Imran Baig. Shealth: A blockchain-based health system with smart contracts capabilities. *IEEE Access*, 8:70030–70043, 2020.

[94] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert Van Renesse. REM: Resource-efficient mining for blockchains. In *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, August 2017. USENIX Association.

[95] Lijing Zhou, Licheng Wang, and Yiru Sun. CP-consensus: a Blockchain Protocol Based on Synchronous Timestamps of Compass Satellite. *IACR*, 2017:1059, 2017.

[96] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.