ACCORD: A Scalable Multi-Leader Consensus Protocol for Healthcare Blockchain

Golam Dastoger Bashar*
Department of Computer Science
Boise State University
golambashar@u.boisestate.edu

Joshua Holmes*
Department of Computer Science
Boise State University
joshholmes@u.boisestate.edu

Gaby G. Dagher
Department of Computer Science
Boise State University
gabydagher@boisestate.edu

***Abstract*—Blockchain is an emerging distributed and decentralized technology that promises to revolutionize the healthcare sector by securely storing and maintaining incorruptible electronic health record data. Consensus protocols are at the core of blockchain technology. They establish security and integrity in the system by ensuring that the majority of miners are in agreement on all transactions and blocks added to the distributed ledger. While many consensus protocols have been proposed, most of them require heavy computation and are not scalable. In this work, we propose a novel permissioned consensus protocol, named ACCORD, a multi-leader (quorum-based) protocol that achieves fork-resistance, robustness, and scalability.**

To achieve this, ACCORD consists of three distinct components: (1) an asynchronous quorum selection procedure to designate the creators of future blocks, (2) a block creation protocol run by the quorum to prevent omissions in the presence of honest quorum members, and (3) a decentralized arbitration protocol to ensure consensus by voting. Additionally, we implemented the protocol and conducted experiments to demonstrate scalability, robustness, and fairness.

I. 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 2018 found that 18% of patients who visited a healthcare provider in the US reported that they had to bring test results to an appointment personally, and 5% needed to have a test or procedure repeated due to the unavailability of prior results [1]. According to the US ONC Health IT [2], in 2019, roughly half of US hospitals were considered interoperable, meaning they are able to efficiently transfer medical records between each other. As the interoperability of hospitals improves, the security of patient records needs to be maintained, as they are valuable targets for cyber attackers.

Hospitals have many methods of storing medical information. According to the CDC, in 2017, approximately 85% of hospitals use some form of EHR[1]. There is no standard for EHR system, so hospitals apply widely varied systems that manage data in significantly different ways [3]. This leads to patient confusion as to where their data is and how to access it [4]. The sharing of EHRs over multiple institutions is a complicated process [5]. This results in poor communication between hospitals. From the patient's point of view, this leads to repeated tests and procedures [6][7], which not only inflates their medical bills but also cause negative patient outcomes.

In a blockchain-based platform, patient records can be represented as a single list of consecutive care events, regardless of where these events occurred [8]. 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 [9]. Therefore, we see it as desirable for the patient to have direct control over access to their EHRs. With the help of blockchain technology, patients can be given control of their health data.

A public permissioned blockchain is a potential solution to the challenges of EHR management. We define a public permissioned blockchain as 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. For a blockchain to be functional and secure, its honest nodes (e.g., hospitals) must not conflict on 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 [10]. 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 hospitals' ledgers, preventing omissions and tampering. As there are around 9000 hospitals and clinics in US[2] and as many as 30 million transactions created per day [11], 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 [12] is the consensus protocol used by the seminal work in blockchain. It allows for a completely permissionless mining system, which is desirable in cryptocurrency. PoW is an extremely energy inefficient protocol, as the miners race to

* These authors contributed equally.

[1]https://www.cdc.gov/nchs/data/nehrs/2017_NEHRS_Web_Table_EHR_State.pdf

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

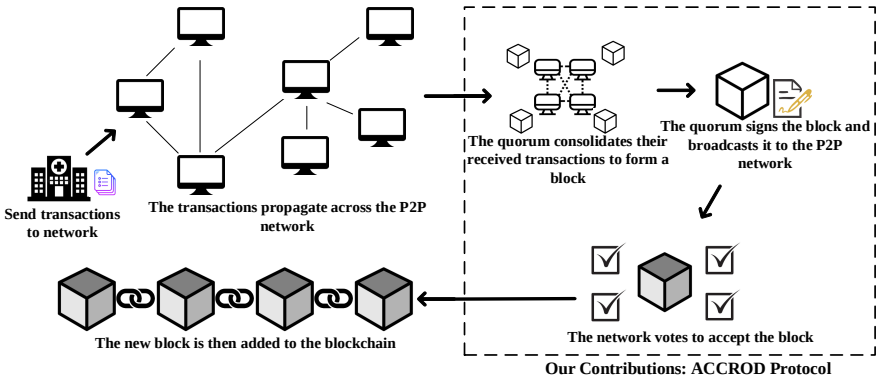


Figure 1: A high level architectural diagram of the system, showing the path a transaction takes to be added to the blockchain.

solve a puzzle to gain the reward that comes with the creation of a block. Security is an issue with PoW, as it can fail if 25% of the computing power is controlled by a malicious party [13][14]. 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.

An alternative paradigm is known as a permissioned consensus protocol, where miners must be certified to mine. A popular permissioned consensus protocol that has been used in the healthcare sector is Practical Byzantine Fault Tolerance (PBFT) [15]. Currently, PBFT is used in some Hyperledger variants [16]. PBFT works efficiently for a small network size. However, due to 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 number of nodes n and messages m increase, the communication burden of this system becomes untenable, at $O(mn^2)$ [17]. The system can continue functioning properly with up to 33% corruption of the mining network.

According to a recent survey [16], 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. Due to the large scale of the healthcare system, strict privacy requirements, and importance of the data, the choice of consensus protocol is important.

Petersen et al. [8] created a consensus protocol for a healthcare blockchain. It attempts to create consensus by having all participating nodes agree on a block with a coordinator. However, the leader selection protocol appears to be open to manipulation and requires a network-wide synchronization. The leader is also capable of ignoring a hospital's transactions.

Three popular leader based consensus protocols are Paxos [18], PBFT [19], and Raft [20]. These systems provide mechanisms to select a leader and allow them to either create or coordinate the creation of a block, which is far more efficient than PoW. However, a single leader is often capable of manipulating the block they are responsible for. Depending on the system, this could allow an adversary to manipulate their block to their advantage. Manipulation techniques could include omission of transactions and manipulation of randomness.

In MedBlock [21], Fan et al. designed a system that attempts to solve the communication overhead of PBFT with large networks. 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 execute PBFT among themselves 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 (e.g., moving the affected nodes out of the region or suspending the malicious nodes).

In this paper, we propose a scalable consensus protocol, named ACCORD that is robust and avoids conflicts over forks. The protocol works as follows: After a transaction is forwarded to the network, ACCORD utilizes a group of leaders called a *quorum*[3] 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 1 is high level view of how data is processed by ACCORD. The main characteristics of the ACCORD protocol are as follows:

- Fork-Resistance[4]: In the presence of competing blocks, if an honest node accepts a block, no other honest node will accept any competing blocks. Either one block is accepted or all are rejected.
- Robustness: In the event of a network outage, the protocol can remain functional or recoverable down to a threshold

[3]Unrelated to Byzantine Quorums [22]

[4]Fork resistance is not immunity to forks.

of operable honest nodes.
- Scalability: The overhead of the protocol should increase at a reasonable rate (e.g. linearly) as the number of nodes and messages increases.
- Liveness: The protocol ensures a valid transaction will appear in all honest node's ledgers within a reasonable period.

To ensure Liveness, the protocol must ensure that a new block will be created by an honest quorum within a reasonable period of time. It is important to note that blocks created by malicious quorums are permissible (if they are valid), as long as honest quorums also regularly create blocks.

In addition, ACCORD achieves fairness in miner selection. ACCORD is fair because it ensures every miner is selected with approximately the same frequency with a more even distribution than random selection. Though ACCORD is designed to meet the needs of a healthcare blockchain, it can be used in blockchain systems for other fields with similar requirements (e.g. supply chain).

II. Related Work

A permissioned blockchain is, by its very nature, more centralized than a permissionless blockchain [23][24]. This centralization allows the nodes in the consensus protocol to have a comprehensive list of miners at all times, therefore allowing the miners to run a more structured consensus protocol. 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 [21][18][19]. The mechanisms behind these protocols vary widely, but they mostly include three major components: leader selection, transaction acquisition, and block creation and distribution. 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 [19][21], but they are often costly, requiring network-wide synchronizations [21]. 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 with an efficient protocol to select the leaders.

Healthcare is naturally centralized around hospitals and requires a certain amount of trust that the data is 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 [5][21][25][26][27], 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 [5][28][29][30] focusing on secure electronic creation, storage, and management of EHRs. Azaria et al. introduced MedRec [28], an Ethereum-based EHR 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. [5] introduced a permissioned blockchain network that uses a cloud-based EHR 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 [29], 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. [30], 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).

In [31], the author's proposed Mneme protocol uses two consensus protocols: Proof-of-Context (PoC) and Proof-of-Equivalence (PoE). PoC is used to store valid blocks of transactions. Their work requires maintenance fees. The concept behind the structure of PoC is to not accept a block as affirmed before a significant percentage of the corroborators is aware of its presence. PoC produces forks, however, so PoE runs periodically and delivers regenesis blocks where a subset of the corroborators is randomly chosen to combine the forks into a unified block.

III. 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 P2P nodes, and external observers.

We assume that the transaction makers honestly create the confidential contents of transactions (e.g. the actual medical records) and do not inappropriately distribute any data. However, we also assume that they are willing to modify the data after the fact if given the opportunity. For example, a doctor may attempt to alter lab results on the blockchain to avoid a malpractice lawsuit. The transaction makers are not trusted to produce the transactions themselves correctly. Therefore, miners must ensure the transactions are valid before adding them to the blockchain.

We assume any given mining node can act maliciously if they are capable of causing damage to the network. Damage to the network includes the following: causing a fork by convincing separate honest nodes of conflicting states in the blockchain, extended outages of the network, and extended periods of malicious control over the mining process. However, if they are unable to cause damage to the network, they are

α	Block Time
β	The block lookback distance for quorum selection
δ	Quorum threshhold for block creation
γ	Priority list threshhold
q	Size of quorum
σ_x	Signature of party x
$\mathcal{B}_n$	Block at position n
$\mathcal{Q}_n$	Quorum selected to mine $\mathcal{B}_n$
N_x	Node with designation x
$\mathcal{SB}_x$	A skeleton block created by quorum member x

Table I: Notation Table

covert[5]. We generally assume that the last party that can apply arbitrary control on a seed in our protocol controls the results of that seed. Due to our quorum selection protocol (discussed in algorithm 1), we assume that malicious nodes are capable of creating blocks that produce 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 P2P nodes that are not mining nodes are considered malicious. P2P nodes, if allowed, require no permission to operate. They are willing to spread disinformation and drop valid transactions if able.

IV. The ACCORD Protocol

A. 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 standard public key K_i in a discrete log cryptosystem on group $\mathbb{G}$[6], and an additive signature key A_i[7]. 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 credentials of all mining nodes are certified by the MSA.

B. Membership Service Authority

Permissioned blockchains differ from permissionless as only authorized nodes are allowed to participate in the mining process. This type 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 keys. 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 IV-E. 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.

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. The MSA could even be the entire mining pool. By distributing the role of MSA across multiple parties, it reduces the required level of trust in any given party. If the MSA is entire 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. 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.

C. Data Propagation

Our system uses a P2P network to transfer information across the network. This information includes transactions, blocks, and votes. A node that receives a valid transaction that has not appeared in a block (an unconfirmed transaction) will add this transaction to their *mempool*. We assume that the P2P network operates correctly with up to 50% corruption of the mining network. Nodes will finish propagating messages across the network in less than target block time α. However, it is not guaranteed that every node will receive every message that propagates, as some nodes may be offline or packets may be lost (e.g., eclipse, DDOS). For example, not every node is expected to receive every block validation or impeachment vote that is produced within α time, resulting in some nodes acquiring a different set of votes after the voting process is complete, but they expect to receive the vast majority of the votes.

A large scale healthcare blockchain would be required to handle an extremely large amount of data. Some estimates are as high as 30 million transactions per day [11]. We

[5]A covert adversary may act maliciously, but are unwilling leave evidence of their malicious action. Their unwillingness to be caught could be due to pressures applied to the party outside the protocol.

[6]Or any cryptosystem that supports the process in Section IV-I

[7]K_i and A_i may be equal if they are from the same cryptographic scheme.

assume that the peer-to-peer network is capable of handling this volume of information as well as the block transmission. If the network is not capable of transmitting this data as fast as it comes in, it will not be able to keep up with all of the data on to the system. We attempt to limit the amount of communication by avoiding the transmission of repeat information. Block transmission, for example, do not have to contain all the transaction information. Instead, it can contain only the transaction headers and structural information, resulting in a block transmission size of approximately 10 MB on average for 30 million transactions. Transaction data would only be sent if the receiving node does not possess a given transaction. Node signatures are also combined using additive signatures, discussed in Section IV-H.

D. Quorum

In a single leader consensus protocol, a leader presents a block to the network. This can cause issues, as either the leader has free rein to alter the block or must validate the block with an expensive set of communications. Additionally, if our protocol is to avoid costly leader elections, the malicious leader may have influence over the selection of future leaders through alterations to their block, allowing a malicious subgroup to potentially 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 determine which outstanding transactions should be included or excluded, each of the quorum members provides 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 2 describes our quorum-based block creation protocol.

Definition IV.1. *Quorum. A quorum is a group of q nodes that is 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 IV-E. Our protocol does not require a specific quorum size q or threshold δ. In our analysis, we found the following q and δ values to be functional given a network size of 10000: $q = 16$, as this allows the quorum size to scale while keeping the quorum size manageable, and $\delta = 13$. These values were determined through experimentation and statistical analysis to produce decent results, as they allow the consensus protocol to reach a balance between robustness against node failure and robustness against malicious quorum members.

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 V). 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 second impeachment, 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 VI-B. We reset δ after a block is successfully created.

E. Quorum selection protocol

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 without the need for a network-wide synchronization while also preventing too much foreknowledge or determinism[8] of the quorum roster.

Our 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[9]. The eligibility of miners is described in Section IV-F. 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 IV-B), while unfortunately allowing malicious parties more time to corrupt quorum members. We recommend $\beta = 7$, as this reduces the effect of certain malicious actions while still limiting the time to coerce malicious action to 70 minutes (assuming a block time $\alpha = 10$ minutes). Algorithm 1 will be used to deterministically select the quorum. Figure 2 illustrates the quorum selection process.

To begin the algorithm, we first set aside the nodes with priority. Any node that has been eligible to mine (e.g., not greylisted) for γ blocks. This priority list, defined by γ blocks since last in greylist, prevents starvation, where a node is not selected to mine for an extended period of time. We define γ as $a \cdot (\frac{n}{q})$. We use $a = 1.5$. The purpose of this priority list is to ensure that all nodes are given the chance to mine. It ensures that the repeated creation malicious quorums cannot consistently select sufficient nodes to create a block.

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

[8]e.g. Algorand [32] would be purely deterministic if the weights or keys never changed

[9]We use exactly two blocks in the miner selection algorithm due to our security assumption that the last quorum with influence over an aspect of the quorum selection process controls the results of that aspect of miner selection algorithm. If the selection process only had one block, that block would fully control the selection of the future quorums, which would allow malicious quorum to select future malicious quorums. More than two blocks would result in greater control by the later blocks rather than a desired equal control across all blocks, as the valid miner pool effects the selection process, which is only an issue with more than two blocks as seeds.

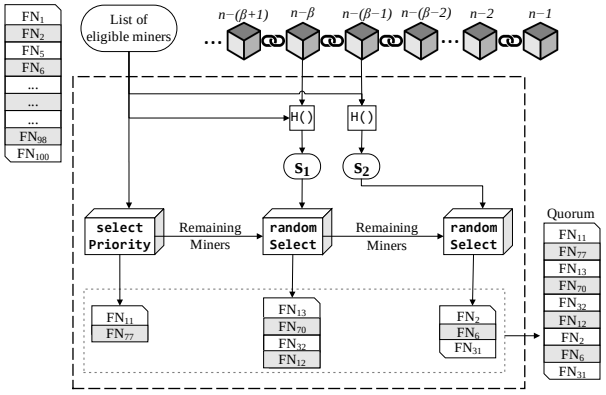


Figure 2: Selection process of quorum members (Algorithm 1).

Algorithm 1 Quorum members selection

To create a quorum to mine block n:
Input: list of x eligible miners $\mathcal{FN}_u = [N_1, N_2, ..., N_k]$, quorum size q, priority threshold γ, and block headers $\mathcal{H}_{n-\beta}$, $\mathcal{H}_{n-(\beta-1)}$ and the number of impeachments making $\mathcal{H}_{n-1}$

1: $s_1 \leftarrow h(\mathcal{H}_{n-\beta}|\mathcal{FN}_u|J)$
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 and reset the list of impeached miners.

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:**
$\mathcal{Q} \leftarrow \mathcal{Q} \cup$ `randomSelect(`$\mathcal{FN}_u$`,` $\left\lceil \frac{q-|\mathcal{Q}|}{2} \right\rceil$`,` s_1`)`
7: s_2 **selects quorum members:**
$\mathcal{Q} \leftarrow \mathcal{Q} \cup$ `randomSelect(`$\mathcal{FN}_u$`,` $q - |\mathcal{Q}|$`,` s_2`)`
8: return $\mathcal{Q}$

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 $(\beta + 2)$th and $(\beta + 1)$th previous blocks, moving back two blocks every time there is an impeachment[10]. This new quorum will include the impeachment as justification as part of their block.

When it comes to fair and efficient leader selection systems, round robin is quite common [33][34]. Every node gets a chance to mine once per cycle. However, a simple round robin opens the system up to several attacks [32]. First of all, a defined order in which the nodes mine can result in the corruption of nodes based on their position in the round robin. Since the order of the nodes is known an indefinite amount of time in advance, adversaries have an indefinite amount of time to mount an attack on nodes to attempt to disrupt the network. A plan could involve engaging in sequential DDOS attacks to target each node as it becomes their turn to mine. The scale and sophistication of attacks can increase given an unlimited amount of foreknowledge. By having future miners mining order be determined randomly, we reduce the ability of adversaries outside the network from harming the integrity of the network.

F. Greylisting

Greylisting is a mechanism to prevent miners from being assigned to mine multiple times in quick succession. When a miner mines a block as a member of a $\mathcal{Q}$, they will be added to the *greylist* and will not be eligible for selection by the Algorithm 1 until they are removed. In the event of impeachment, $\mathcal{Q}$ failed to mine a block and an alternate $\mathcal{Q}'$ succeeded. To prevent manipulation of the quorum selection process, $\mathcal{Q}'$ will be added to the greylist after β blocks and are eligible to mine until that time. $\mathcal{Q}$ will be ineligible to mine for β blocks, but will not be treated as having entered the greylist for the purposes of Algorithm 1. 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 if the list exceeds 33% of the nodes, following a FIFO structure. Greylisted nodes should still sign to accept blocks, propagate blocks, and 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 theoretically have the ability to select the future quorum. If the malicious party gets lucky enough to randomly gain control of β consecutive blocks, this could lead continually selecting their own members for mining duty. The greylist helps prevent this by reducing the number of available cooperating malicious nodes. These nodes would need to be able to saturate the greylist to maintain control. Another advantage of greylisting is to allow nodes to have a period of time where they do not need to be active. For example, after a quorum member mines and is greylisted, they can perform maintenance on their system, knowing they will not be selected to a quorum in the near future.

G. 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 block header $\mathcal{H}$ 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 3 illustrates the structure of a block.

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

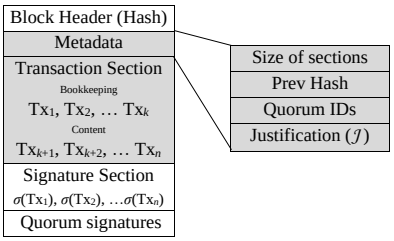


Figure 3: Block structure. The grey portions will be used in the Quorum Selection Protocol (Algorithm 1).

The transaction section includes two types of transactions: the MSA bookkeeping transactions and content transactions. Included among the bookkeeping transactions are the quorum members' *null transactions*, discussed in Section IV-I. The bookkeeping is expected to be small, possibly only consisting of the null transactions. The content transactions will generally be the largest section, potentially containing thousands of transactions per block. All transactions are sorted by transaction header within their sections. The signature section will include the signatures of the transactions in the same order or structure that they appear in the transaction section. These transactions are also in a Merkle Tree [35], allowing for faster verification of a transaction's existence in a block.

In this blockchain, there is no defined size limit imposed on the blocks, as every transaction must be added to the blockchain eventually. Quorum members attempt to add every valid transaction from their mempool to the block. We assume that the peer-to-peer network can handle the throughput of raw transactions, so the transmission size of the block must be of reasonable size. Assuming 30 million transactions in a day, if only block skeletons are transmitted, then the block transmission will usually be less than 10 MB. Extended block creation times due to impeachment can result in larger blocks. These should never exceed 200 MB given the protections against node fault tolerance (see Sections VI-B and VII-D for statistics on impeachment). Given Bitcoin has a median propagation of 12.6 seconds to propagate 1 MB blocks [36], a 10 minute propagation time should be more than adequate.

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 empty blocks except that they are created by the MSA.

A block is valid if (1) all contained transactions are valid, (2) the block structure is correct, (3) the metadata is accurate, and (4) the block is signed by a valid quorum.

H. 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. Any arbitrary non-cooperative aggregate signature scheme will work, so we will employ the signature scheme created by Boneh et.al. [37]. To use this scheme (or similar schemes), each signed message should be unique, so the messages can be appended with K_i, their standard public key. Node N_i will sign these messages using A_i.

I. Null Transaction

In ACCORD, 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. Null transaction must be infeasible to forge without the node's private key. This is accomplished using a key image technique [38][39]. In this approach, we add an aspect of the previous block to the public portion of the generator creation hash to ensure a unique key image for each node on each block. Let K_i be quorum member i's standard public key, s_i be their private key, and $H_{\mathcal{G}}$ be a hash-to-group function. Then, quorum member i's key image I_i is defined as follows:

$$I_i = s_i \cdot H_{\mathcal{G}}(K_i | \mathcal{H}_{n-1}) \tag{1}$$

Each party's null transaction will only contain I_i and proof that I_i is well-formed given the party's credentials. To prove this is well-formed, we will use a Zero-Knowledge Proof to prove that $((\mathbb{G}, G, H_p(K_i|\mathcal{H}_{n-1}), K_i, I_i), s_i)$ is a Diffie-Hellman tuple[11] [40]. This proof is made non-interactive using the Fiat-Shamir heuristic [41]. 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.

J. 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 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 IV-K.

K. 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

[11] e.g, in the Diffie-Hellman key exchange, $((G, A, B, K), b)$, where $a \cdot G = A$, $b \cdot G = B$, and $a \cdot b \cdot G = K$, would be a Diffie-Hellman tuple, as the relations $b \cdot G = B$ and $b \cdot A = K$ share the same key b.

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.

L. 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 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 (e.g. $\frac{\alpha}{5}$), 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 sent from other quorums. This is also called `requesting`.

M. 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 results of Algorithm 1 can be verified by any node wishing to verify that this set of nodes may create the block. $\mathcal{Q}_n$ is expected to create a valid block within α time of receiving the previous block. 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. Note that if any P_j fails to respond through any accepted channel within a prescribed time, they will be treated as absent and omitted from the protocol. After this, P_i opens their commitment to each P_j, sending them $\mathcal{SB}_i$ and waiting for the prescribed 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 (defined in Section IV-I) 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 each 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[12], treating the quorum list as cyclical[13]. 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 IV-G. 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 the block as stated in Section IV-G. 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 broadcast $\mathcal{B}_n$ to the P2P network. $\mathcal{B}_n$ is first transmitted as a block skeleton with metadata. If the receiving node does not possess a transaction in $\mathcal{B}_n$, they will request it from the sending node. $\mathcal{B}_n$ will be 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 $\mathcal{Q}'_{n+1}$ is the quorum responsible to mine if $\mathcal{Q}_{n+1}$ 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 $\mathcal{B}_{n-1}$.

N. Synchronicity Model

While we aim to produce an asynchronous block creation process, ACCORD as a whole is partially synchronous, in that we assume that the majority of messages arrive at their destination within a time, but some some messages may be blocked by an adversary. We aim to avoid any synchronizations between the creation of two blocks, but the network-wide validation of the block requires a vote, which is a partial synchronization.

The process of selecting a quorum $\mathcal{Q}_n$, creating a block $\mathcal{B}_n$, publishing it, selecting a new quorum $\mathcal{Q}_{n+1}$, and creating the block $\mathcal{B}_{n+1}$ requires no network-wide synchronization. However, it does require the quorums involved to be fully synchronous. For impeachment and block validation, the voting process is partially synchronous, as the votes collected by individual nodes do not have to be consistent for an impeachment to be successful.

V. Mining rules

A. 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 V.1. *Pending: $\mathcal{B}$ is pending if it has neither been accepted nor abandoned.*

Definition V.2. *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 $\mathcal{B}$.*

Definition V.3. *Abandoned: $\mathcal{B}$ from $\mathcal{Q}$ is abandoned if:*

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

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

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

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.

1: **Initialization.** P_i creates a skeleton block $\mathcal{SB}_i$ with all available transaction headers from their mempool.
2: `broadcast`$(P_i, \mathcal{Q}, \mathcal{SB}_i)$ // P_i exchanges $\mathcal{SB}_i$ with all P_j
3: `request`$(P_i, \mathcal{Q}, \mathcal{SB}_j)$
4: P_i creates their null transaction I_i.
5: `broadcast`$(P_i, \mathcal{Q}, I_i)$
6: `request`$(P_i, \mathcal{Q}, I_j)$
7: P_i verifies the validity of all I_j. If I_j is invalid, P_j are omitted from the mining process.
8: P_i creates the skeleton master block $\mathcal{SMB} \leftarrow \bigcup_{l \leftarrow 1}^{q} \mathcal{SB}_l$
9: **for all** transactions $\mathcal{T}_x \in \mathcal{SMB} - \mathcal{SB}_i$ **do** // Get unknown Txs
10: `request`$(P_i, \mathcal{Q}, \mathcal{T}_x)$
11: P_i creates the master block $\mathcal{MB}$
12: $\mathcal{H}'_n \leftarrow$ `block_hash`$(\mathcal{MB}_i)$ // P_i creates block header
13: $\mathcal{MH}_i \leftarrow$ `full_block_hash`$(\mathcal{MB}_i)$ // P_i creates hash to sign
14: `broadcast`$(P_i, \mathcal{Q}, \mathcal{MH}_i)$
15: `request`$(P_i, \mathcal{Q}, \mathcal{MH}_j)$
16: **let** $z \leftarrow$ the size of the largest group in $\mathcal{Q}$ broadcasting the same $\mathcal{MH}$
17: **if** $z = q$ **then** // All P_j created the same $\mathcal{MH}_j$
18: $\mathcal{B}_n \leftarrow \mathcal{MB}_i$
19: $\mathcal{H}_n \leftarrow \mathcal{H}'_n$
20: **else if** $z < q$ **then**
21: `broadcast`$(P_i, \mathcal{Q}, \mathcal{MB}_i)$
22: `request`$(P_i, \mathcal{Q}, \mathcal{MB}_j)$
23: **while** $t_{\text{elapsed}} < t_{\text{limit}}$ or $z < \delta$ **do**
24: Attempt to reconcile differences using message log to create $\mathcal{B}_n$.
25: **if** $(\mathcal{MB}_i = \mathcal{B}_n)$ // If P_i agrees with $\mathcal{B}_n$, P_i signs the block.
26: σ_i = `block_sign`$_{P_i}(\mathcal{MB}_i)$
27: `broadcast`$(P_i, \mathcal{Q}, \sigma_i)$
28: `request`$(P_i, \mathcal{Q}, \sigma_j)$
29: **let** $z' \leftarrow$ the number of valid signatures of $\mathcal{MH}$
30: **if** $(z' \geq \delta)$
31: Add all σ_j to $\mathcal{B}_n$
32: **return** $\mathcal{B}_n$
33: **else goto** step 28.

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

Definition V.4. *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.*[14]

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*.

B. 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.

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

1) 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 $M_A \leftarrow$ (`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.

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.

2) 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 ($\mathcal{B}^1_n$, $\mathcal{B}^2_n$...) 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 $\mathcal{B}_n$.

In the event of impeachment, N_i will create a message $M_I =$ (`impeachment_flag`$|\mathcal{MH}_{n-1}|c|K_i$), 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 A_i is N_i's additive public key. M_I is then signed and broadcasted to the network. Quorum members can vote for their own impeachment. Impeachment

votes should be accompanied by the motivation to impeach (e.g. block not created, multiple blocks, invalid block, etc).

If $\mathcal{B}_n$ gets accepted and the impeachment on $\mathcal{Q}_n$ 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.

C. 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 $\mathcal{Q}_n$. If a quorum were to release $\mathcal{B}_n$ and $\mathcal{B}'_n$ some time later, then the vast majority of honest nodes will detect $\mathcal{B}_n$ first. It is possible that some honest nodes will also sign $\mathcal{B}_n$. However, it is highly unlikely that any honest node would sign $\mathcal{B}'_n$, as if they receive $\mathcal{B}'_n$ first, it is highly unlikely for them to not see $\mathcal{B}_n$ or evidence of $\mathcal{B}_n$'s existence (e.g. impeachment votes) before α time has passed.

However, it is theoretically possible that multiple blocks could both achieve a majority. If honest nodes' votes are divided 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, it will vote to impeach $\mathcal{Q}_n$, rejecting all $\mathcal{Q}_n$'s blocks unless one of them is *well-established*. Such an attack is unlikely, as it would require a segregation of the P2P network.

VI. 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 to control the mining process, and miner selection distribution. These experiments are purely statistical and are hardware agnostic.

A. Communication costs

To determine the communication overhead of our mining process, we measured 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%, as not all transactions will have propagated to every node in the quorum or the network. 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.

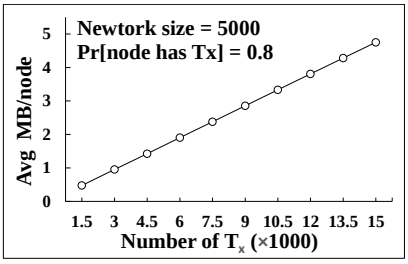
Figures 4[a-c] shows the communication overhead of the exchange of skeleton blocks in MB. Figure 4(a) shows that as the number of transactions increases, the size of the communications associated with the exchange of skeleton blocks scales linearly. In Figure 4(b) we hold the number of transactions constant. 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 4(c), the trends seen in Figure 4(a) and Figure 4(b) are apparent and there does not appear to be any emergent complexity. Figures 5[a-c] show similar trends, but the scaling of these graphs shows these communications are negligible compared to the skeleton blocks.

B. Fault Tolerance

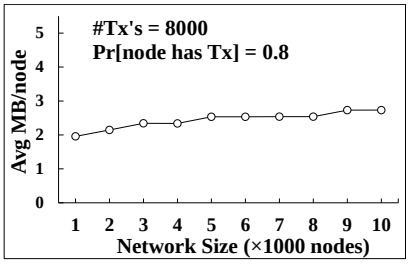
These fault tolerance experiments show the robustness of our consensus protocol against different percentages of inactive nodes. We measure robustness by measuring average number of impeachments with different signing thresholds, assuming a maximum propagation time $\alpha = 10$ minutes. We assume that the nodes that are active in the network are behaving honestly in this context. Here we consider 15 nodes in the quorum and 10000 mining nodes in the network.

Figure 6 with a varying number of nodes initially required to sign a block. In this setup, if the next block has 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 examining the creation of 10 consecutive block, as most outages are resolved fairly promptly. If there is an extended outage, the MSA can temporarily suspend the effected nodes withing 10 blocks.

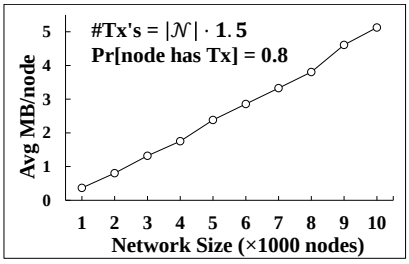
Figure 6(a) shows how the network responds to small outages of less than 10%. If we required all 15 quorum to sign a block, the average block time increases to untenable levels very quickly, taking over an hour with a 10% network outage. The average time is significantly improved by requiring only 14 or 13 quorum signatures, reducing the average block



(a) Scaling Tx

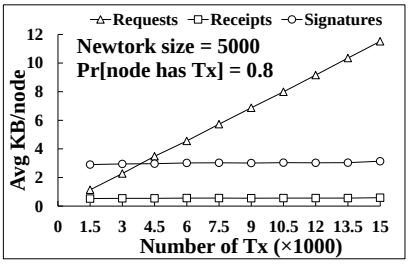


(b) Scaling network size

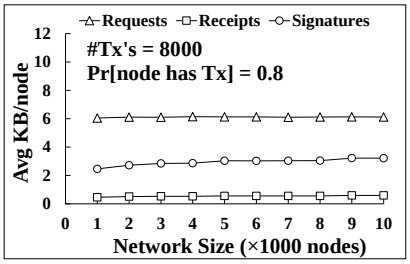


(c) Scaling network size and Tx

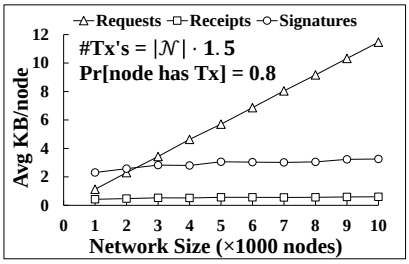
Figure 4: Average communication overhead per node from skeleton blocks with different number of transactions and nodes.



(a) Scaling Tx



(b) Scaling network size



(c) Scaling network size and Tx

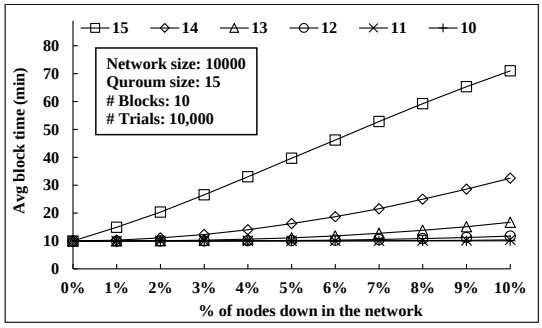
Figure 5: Average communication overhead per node of smaller messages within quorum to build a Master block.

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 cause security concerns, discussed in Section VI-C. Figure 6(b) shows the recoverability of the system given larger outages of up to $50\% - 1$ of the network. 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.

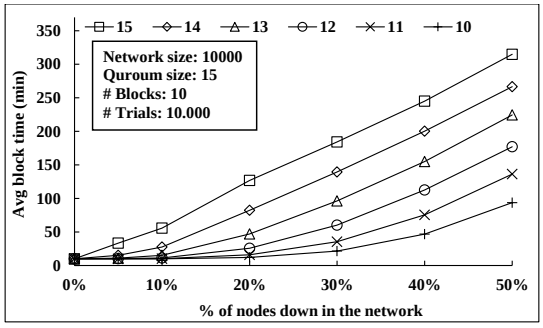
C. Manipulation in selection

A common concern with miner selection schemes use previous blocks to select future miners is the potential that a malicious block can select malicious miners, which could allow these malicious miners to take control of the system by repeatedly selecting other malicious miners. With this experiment, we intend to show that our protocol is resistant to such an attack. This attack is discussed in more detail in Section VII-C.

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 a quorum is malicious, they can create a block that will produce a selection of nodes of their choosing. In this experiment, we assume that the malicious nodes will not use impeachment to maintain control, as this is a detectable behavior. 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



(a) Blocktime up to 10% nodes unavailable



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

Figure 6: Average block time given increasing network outages. $\alpha = 10$ minutes.

settings, similar to the experiment in subsection VI-B. Finally, we compared our two seed model with an equivalent single

seed model. Our test involved selecting 100,000 quorums, and we ran this test 5 times and averaged the results.

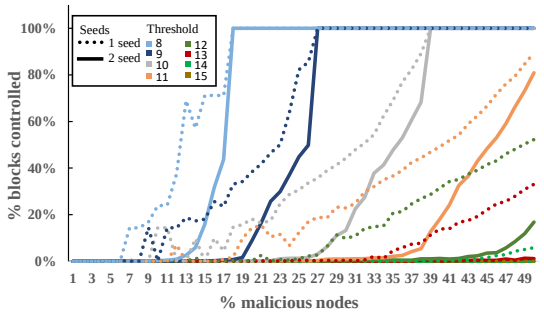
The strategy employed by the different versions of this protocol are as follows: On the single seed version, if the $(n-\beta)$th quorum is malicious, they will attempt to make the nth quorum malicious with the minimum possible number of malicious nodes. They will prioritize malicious nodes that have 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. If there are too many high prioirty honest nodes or too few malicious nodes are available, they will be unable to take control of quorum n and will instead fill the quorum with only higher priority honest node to improve the probability that their malicious group will regain control in the future.

On the two seed version, a single quorum has less control, needing control of the $n-\beta$ and $n-(\beta-1)$ quorums to control quorum n. In all cases, if there are too many high priority honest nodes or not enough eligible malicious nodes to take control, they will fill all slots with higher priority honest nodes. If they have control of both quorums, they will follow the single seeded approach. If they control the $(n-\beta)$ quorum but not the $n-(\beta-1)$ quorum, they will fill their half of the selection with lower priority malicious nodes to increase the probability that block n will be malicious. 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

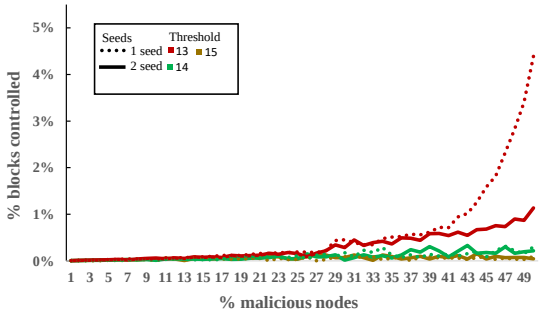
Figure 7(a) shows that requiring too low of a threshold makes the protocol too vulnerable to this attack, with the worst case of a threshold of 9 in the quorum causing the honest nodes to lose control at 15% malicious nodes. The results improve as the threshold increases, requiring that the malicious group use more of their membership per block and reducing their ability to prevent honest nodes from attaining priority. Figure 7(a) demonstrates that the two seed version generally reduces malicious control of the network up to a breaking point, where both versions reach 100% control at the same level of corruption.

On Figure 7(b), we narrow the y-axis to a maximum of 5%, removing the lines that exceed that value 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 lower thresholds, whereas its single seed counterparts have approximately 4% of the blocks created being created by the malicious group. This, combined with our robustness experiments in Subsection VI-B, 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 single seeded version allows for the malicious group to gain control of sections, whereas the two seeded version requires the malicious group to randomly gain control of β quorums in a row.



(a) All thresholds



(b) Trimmed thresholds

Figure 7: 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.

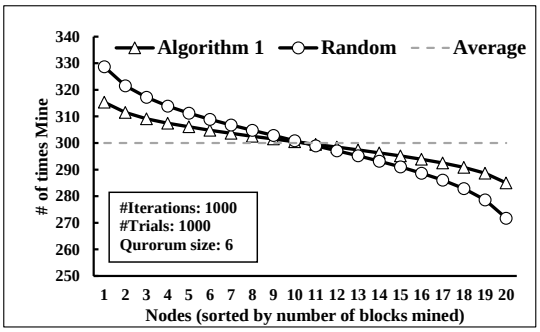
D. Miner Selection Distribution

To quantitatively analyze the models in terms of fairness, we 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.

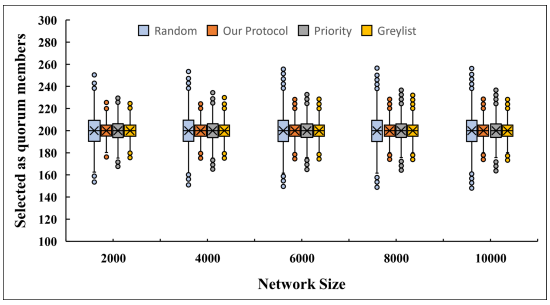
Figure 8(a) reported below is showing that the distribution of our protocol and a random selection. The x-axis represents the nodes based on their rank, meaning they are sorted by the number of times they mined. The y-axis indicates the number of times these nodes mined. We ran this experiment 1000 times, 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.

Figure 8(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 8(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.



(a) Frequency of quorum selection with 20 nodes.



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

Figure 8: Distribution of quorum selection

VII. Threat-Risk Assessment Model

Consensus protocols have some common vulnerabilities that need to be addressed. Below, we highlight a few of these and our mitigation techniques against these vulnerabilities.

A. 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 2) 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 V-B1) 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 V-B2) 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.

B. Long-Range attack

The Long-Range attack is a common vulnerability in consensus protocols aiming to be energy efficient. In the Long-Range attack, an adversary gains control of a majority of the mining network at a certain block in the past. This allows them to unilaterally create a fork from that point by using the stolen deprecated credentials to execute the consensus protocol.

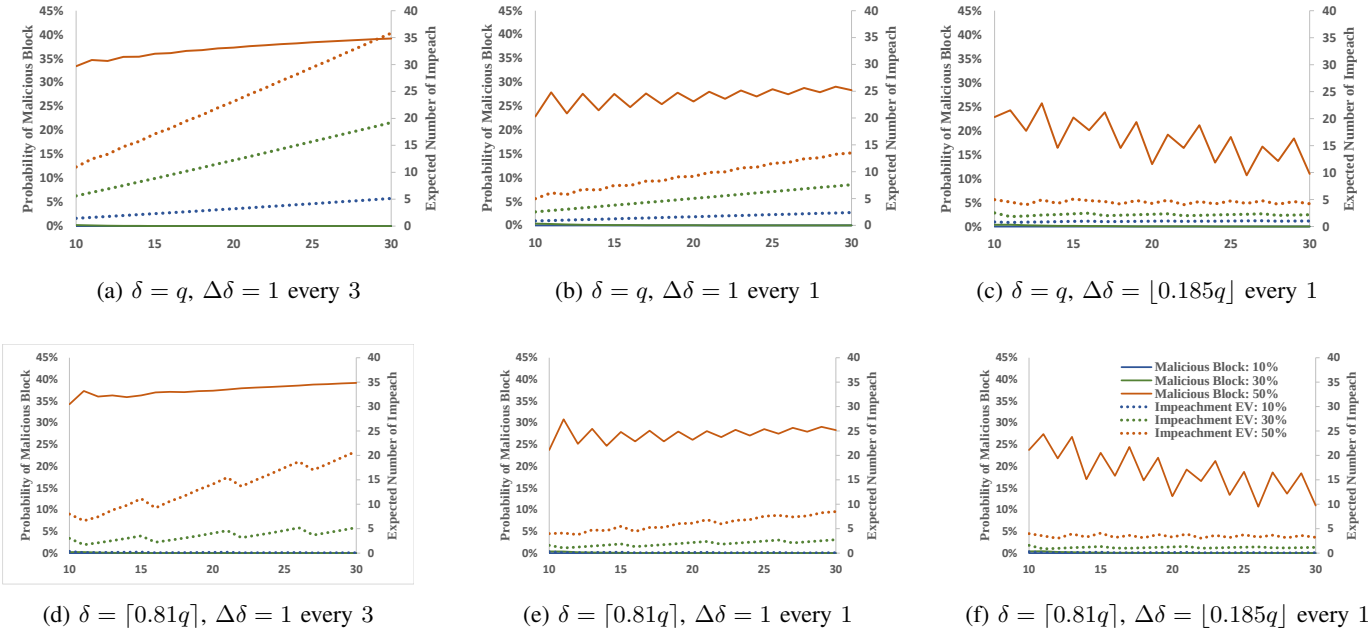
Moving checkpoints is a mitigation approach utilized by many PoS based protocols. The concept behind moving checkpoints is that there is a quota on the most delinquent n number of blocks of the chain which can be reorganized (e.g., the quota for Peercoin [42] is restricted to one month's worth of blocks and for NXT it is of a few days or hours. In our system, this attack could be executed if the adversary acquired enough private keys from miners who were members of the mining pool at a block in the distant. This gathering of keys could take place over the course of years and use keys that were depreciated in the past. If the attacker accumulates the majority of the private keys of the network at that block, they may rerun the consensus protocol and write a new history of the blockchain from that point.

Our protocol is vulnerable to Long Range Attack, as there is no concrete mechanism to prove which branch is valid to someone validating from the genesis block. 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. However, such a branch would be fairly identifiable at the point of divergence, as there would likely be a purge or mass re-keying of mining nodes or a significant number of impeachments on the fake branch. Additionally, since this protocol is nominally designed for healthcare, an observer can ask hospitals which branch is real, which they can show by providing transactions signed by them on the appropriate chain.

C. Future miners selection attack

D. Statistical Analysis of Impeachment and Malicious Block Takeover

Experiment VI-C examines a strategy to maintain control of the mining process assuming no impeachment. While repeated



(a) $\delta = q$, $\Delta\delta = 1$ every 3 (b) $\delta = q$, $\Delta\delta = 1$ every 1 (c) $\delta = q$, $\Delta\delta = \lfloor 0.185q \rfloor$ every 1

(d) $\delta = \lceil 0.81q \rceil$, $\Delta\delta = 1$ every 3 (e) $\delta = \lceil 0.81q \rceil$, $\Delta\delta = 1$ every 1 (f) $\delta = \lceil 0.81q \rceil$, $\Delta\delta = \lfloor 0.185q \rfloor$ every 1

Figure 9: Comparison of Probabilities of either malicious quorum with impeachment or perpetual impeachment at different levels of malicious control.

impeachment may be infeasible to maintain control of the system due to the suspicious activity required to achieve it, it may be considered a viable strategy.

First of all, we define n as the number of available nodes, m as the number of malicious nodes, δ to be the original threshold, δ_i to be the threshold after i impeachments, $\Delta\delta$ as the number of impeachments required to decrease the threshold and the amount the threshold is decreased. For the sake of simplicity, we ignore the ineligibility of nodes after impeachment, as the proportions of nodes are unlikely to change significantly before the block is finalized. We also assume that any node is as probable to be in the greylist as any other node.

To analyze the probability of a malicious subgroup gaining control of a single block, we created a probability model of our protocol assuming random seeds. To begin, we must produce the probability of creating a malicious block. $Pr[\text{mal by } \infty] = \sum_{i=0}^{\infty} Pr[\text{mal at } i] \cdot Pr[\text{imp to } i]$, as to achieve a malicious block after i impeachments, the malicious group must get to that level of impeachment and get enough of their cohorts in the resulting quorum to achieve a malicious block. $Pr[\text{mal at } i] = \frac{\sum_{i=\delta_i}^{q} \binom{i}{m}\binom{q-i}{n-m}}{\binom{q}{n}}$. The probability that a malicious group is unable to take malicious able to cause an impeachment $Pr[\text{imp of } i] = \frac{\sum_{i=q-\delta_i+1}^{\delta_i-1} \binom{i}{m}\binom{q-i}{n-m}}{\binom{q}{n}}$, as the adversary have at enough nodes to cause an impeachment, but not enough to control the quorum The probability of achieving an impeachment level of i without hitting a malicious quorum is $Pr[\text{imp to } i] = \prod_{j=0}^{i-1} Pr[\text{imp of } i]$.

Alternatively, a malicious subgroup may attempt to cause the system to prevent the creation of new blocks rather than attempting to control the block creation process. In this context, the adversary is not concerned with controlling blocks. Instead, they are solely focused on impeachment. $Pr[\text{imp to } i] = Pr[\text{imp to } i-1] \cdot \frac{\sum_{i=q-\delta_i+1}^{q} \binom{i}{m}\binom{q-i}{n-m}}{\binom{q}{n}}$.

Using the recommended settings of $q = 16$, $\delta = 13$, $\Delta\delta = 2$ every 2, the probability of a malicious group being able to take control of any given block at 50% corruption is 23.2%. The probability of the malicious group taking control of 7 sequential blocks is 0.0074%.

Using the probabilities, we can examine the relationship between the variables, including q, δ, and $\Delta\delta$, shown in Figure 9. These graphs show some rather interesting trends. First of all, it shows that at high levels of corruption, having the threshold δ decreases the probability of malicious control. Predictably, having a smaller δ also decreases the number of impeachments. Additionally, the smaller initial δ does not have a significantly higher probability of malicious control as the quorum size increases. In regards to impeachment, having a low $\Delta\delta$ results in unacceptable impeachment levels that increase as the quorum size increases. Having a high $\Delta\delta$ that scales with q appears to result in generally better statistics as the quorum size increases. One notable aspect is the jagged lines on (b), (c), (e), and (f). This is likely caused by the fact that odd quorums have, proportionally, a higher minimum δ, $\lceil q/2 \rceil + 1$. It appears that being able to reduce δ to a lower value reduces the probability of the malicious party gaining control the quorum. From Experiment VI-C, we know that too low of an initial δ can result in more susceptibility to a malicious takeover, so it is important to find a balance.

It is important to note that the use of impeachment to take control over a block is a detectable malicious action, therefore

the malicious subgroup would want to use it sparingly, as the network would likely remove the malicious nodes who participated.

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. 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 nodes 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 that the adversary can precisely choose the future quorum members that the block will select. This alone does not give the adversary the ability to corrupt a future quorum, as block $\mathcal{B}_n$ selects half of each quorum $\mathcal{B}_{n+(\beta-1)}$ and $\mathcal{B}_{n+\beta}$. To control a future quorum, the adversary needs two consecutive blocks or randomly have an opportunity to gain control through random chance or impeachment. To theoretically control every block, they need to control β consecutive blocks. If they they control less than β blocks, then every β blocks, they will choose the the roster of one fewer quorums. 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. 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 few days. In our experiments, the malicious subgroup never gained controlled β consecutive quorums.

VIII. Security Analysis

This section 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 $\mathcal{H}_n$. These proofs show that partial malicious members of a quorum $q' < \delta$ can not manipulate quorum members selection means they can only create a block or force an impeachment (if $q' > q - \delta$).

Proposition 1. *If a selected quorum $\mathcal{Q}_n$ with malicious members $q' < \delta$ broadcasts a valid block $\mathcal{B}_n$ with 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 members $q' < \delta$ 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 members $q' < \delta$ 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 $\sum_{k=q-\delta}^{q} \binom{q'}{q-k}$.

IX. Conclusion and Future Work

In this work, we have presented ACCORD, 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.

As for future work, our protocol 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. It would also be interesting to explore methods to select future quorums without it being dependent on the list of transactions while continuing to prevent indefinite foreknowledge of quorum membership. This would negate the possibility of a malicious takeover of the mining process as discussed in Experiment VI-C.

REFERENCES

[1] "Gaps in individuals' information exchange," Jul 2021. [Online]. Available: https://www.healthit.gov/data/quickstats/gaps-individuals-information-exchange

[2] C. Johnson and Y. Pylypchuk, "Use of certified health it and methods to enable interoperability by us non federal acute care hospitals," 2021.

[3] "Why EHR data interoperability is such a mess," Oct 2018, [Online; accessed 9. Feb. 2021]. [Online]. Available: https://www.healthcareitnews.com/news/why-ehr-data-interoperability-such-mess-3-charts

[4] K. Caine, S. Kohn, C. Lawrence, R. Hanania, E. M. Meslin, and W. M. Tierney, "Designing a patient-centered user interface for access decisions about ehr data: implications from patient interviews," *Journal of general internal medicine*, vol. 30, no. 1, pp. 7–16, 2015.

[5] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," in *AMIA annual symposium proceedings*, vol. 2017.

[6] F. Greer, C. McLean, and T. Graham, "Caffeine, performance, and metabolism during repeated wingate exercise tests," *Journal of applied physiology*, vol. 85, no. 4, pp. 1502–1508, 1998.

[7] J. Waller, K. McCaffery, H. Kitchener, J. Nazroo, and J. 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*, 2007.

[8] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proc. NIST Workshop Blockchain Healthcare*, vol. 1, no. 1, 2016, pp. 1–10.

[9] "HHS Rules to Health Data," 2020. [Online]. Available: https://www.hhs.gov/about/news/2020/03/09/hhs-finalizes-historic-rules-to-provide-patients-more-control-of-their-health-data.html

[10] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.

[11] "Change Healthcare: blockchain with 30M tpd," 2019, [Online; accessed 28. Jan. 2021]. [Online]. Available: https://www.ledgerinsights.com/change-healthcare-enterprise-blockchain-with-30-million-transactions-per-day

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

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

[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)*, 2019, pp. 190–195.

[15] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th SRDS*.

[16] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, and A. Faxvaag, "Blockchain in healthcare and health sciences—a scoping review," *International Journal of Medical Informatics*, vol. 134, p. 104040, 2020.

[17] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, 12 2020.

[18] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[19] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[20] T. Nakagawa and N. Hayashibara, "Energy efficient raft consensus algorithm," in *International Conference on Network-Based Information Systems*. Springer, 2017, pp. 719–727.

[21] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, p. 136, 2018.

[22] D. Malkhi and M. Reiter, "Byzantine quorum systems," *Distributed computing*, vol. 11, no. 4, pp. 203–213, 1998.

[23] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018.

[24] G. D. Bashar, A. A. Avila, and G. G. Dagher, "Poq: A consensus protocol for private blockchains using intel sgx," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2020.

[25] X. Liu, Z. Wang, C. Jin, F. Li, and G. Li, "A blockchain-based medical data sharing and protection scheme," *IEEE Access*, vol. 7, 2019.

[26] F. Tang, S. Ma, Y. Xiang, and C. Lin, "An efficient authentication scheme for blockchain-based electronic health records," *IEEE access*, 2019.

[27] M. Zghaibeh, U. Farooq, N. U. Hasan, and I. Baig, "Shealth: A blockchain-based health system with smart contracts capabilities," *IEEE Access*, vol. 8, pp. 70 030–70 043, 2020.

[28] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *2016 IEEE 2nd International Conference on OBD*.

[29] D. 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, US*, 2016.

[30] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE S&P Workshops*.

[31] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Mneme: A mobile distributed ledger," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1897–1906.

[32] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.

[33] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[34] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[35] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," *Ruhr-University Bochum, Tech. Rep*, vol. 12, p. 19, 2008.

[36] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*, 2013, pp. 1–10.

[37] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Theory and Applications of Cryptographic Techniques*. Springer, 2003.

[38] J. K. Liu, V. K.-W. Wei, and D. S. Wong, "Linkable and anonymous signature for ad hoc groups," 2004.

[39] N. Van Saberhagen, "Cryptonote v 2.0," 2013.

[40] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols: Techniques and Constructions*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[41] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[42] S. King and S. Nadal, "Peercoin–secure & sustainable cryptocoin," *Aug-2012 [Online]. Available: https://peercoin.net/whitepaper*, 2012.