Order-Fair Consensus in the Permissionless Setting

Mahimna Kelkar* Soubhik Deb† Sreeram Kannan†

February 8, 2021

Abstract

Over the past five years, a significant line of research has investigated the blockchain consensus problem in the general permissionless setting, where protocol nodes can leave and join dynamically. The work of Garay et al. (Eurocrypt 2015) and Pass et al. (Eurocrypt 2017) showed the security properties of *consistency* and *liveness* for Nakamoto's seminal proof-of-work protocol. However, consistency and liveness do not provide any guarantees on the relationship between the order in which transactions arrive into the network and the finalized order in the ledger, making protocols prone to transaction order-manipulation attacks. As a solution, a recent paper by Kelkar et al. (Crypto 2020) introduced a third useful property for consensus protocols: *transaction-order-fairness*. Their model was limited to the classical (permissioned) setting, where the set of protocol nodes is fixed a priori, and does not fit well for permissionless environments where order-manipulation attacks have been most prominent.

In this work, we initiate the investigation of order-fairness in the permissionless setting and provide two protocols that realize it. Our protocols work in a synchronous network and use an underlying longest-chain blockchain. As an added contribution, we show that any fair ordering protocol achieves a powerful zero-block confirmation property, through which honest transactions can be securely confirmed *even before they are included in any block*.

*Cornell University and Cornell Tech. mahimna@cs.cornell.edu

†University of Washington, Seattle.

Contents

1 Introduction

The state machine replication, or blockchain consensus[1] abstraction pertains to protocols where a set of players seek to agree on an ever-growing, linearly-ordered log of transactions. Historically, in the classical or permissioned setting, where the set of players is known a priori, security was characterized in terms of two properties: consistency and liveness. *Consistency* ensures that all nodes have the same view of the log, while *liveness* ensures that new transactions input to the system get added to the log soon. Modern permissionless protocols, operating for example on proofs-of-work [32], have been shown to realize the same consistency and liveness properties, but in a new setting, where the set of players is dynamic and unknown [22, 33].

Unfortunately, neither consistency nor liveness, enforces any guarantees on the relationship between the order in which transactions arrive into the network and the final transaction ordering in the log. In existing blockchain protocols, both permissioned and permissionless, an adversary can significantly influence the final transaction ordering. This problem is exacerbated in permissionless protocols, which have significant latency between the time at which a transaction is first seen and the time at which it gets confirmed (eg., the confirmation latency of Bitcoin is in the order of hours). During this window, an adversary can manipulate the transaction order as well as introduce new competing transactions ahead of the older honest transaction into the log, thus *front-running* it. For example, in blockchains like Bitcoin and Ethereum, an adversarial miner has complete control over the inclusion and ordering of transactions in the block that it mines. The lack of a fair transaction ordering property is especially problematic for decentralized financial systems where transaction ordering is highly relevant and can impact the profitability of a transaction. In particular, severe real world consequences of order-manipulation attacks have been shown by Daian et al. [15] and Eskandari et al. [17].

As a solution, recent work by Kelkar, Zhang, Goldfeder, and Juels [26] (henceforth KZGJ) proposed a new consensus property, called *order-fairness*. Informally, if a transaction tx_1 was received before another one tx_2 by a large fraction of the nodes in the network, order-fairness dictates that tx_1 be sequenced in the output log first. KZGJ was the first to formalize a fair transaction ordering property for the consensus abstraction and provide protocols to realize it in both synchronous and asynchronous network models. However, KZGJ only considered the permissioned setting. Consequently, the protocols do not directly apply to the attacks from [15, 17], which were exclusively on permissionless blockchains.

Importance of fair transaction ordering. Decentralized finance, or DeFi, has become massively popular on permissionless blockchains in the last few years. As of January 2021, Ethereum houses more than 20 billion USD [1] of locked capital in DeFi smart contracts. Popular DeFi exchanges like Uniswap V2, Compound Finance, and Curve each routinely handle 100 to 200 million USD in transaction volume every day [2].

Transaction ordering is crucially important for DeFi applications, as the execution order can determine the validity and profitability of a transaction. In many scenarios, *when* a transaction is executed is far more important than *whether* it was executed at all. Execution in an unfavorable order can render a transaction useless to a user. In the context of decentralized exchanges, which allow users to trade between different cryptocurrencies, the execution order of a transaction can also influence the exchange price the user gets.

[1] We use the terms state machine replication and consensus interchangeably.

The ease of ordering manipulation in permissionless protocols can cause high instability in DeFi applications. Indeed, as Daian et al. [15] found, dozens of bots sat waiting on the Ethereum network to profit by adversarially manipulating the execution order of ordinary user transactions. During the initial coin offerings (ICOs) of the Status.im and BZRX tokens, adversaries were able to manipulate transaction ordering to get a better price to buy the tokens [3, 17].

This is not unlike the impact of high-frequency-trading on Wall Street [29], before regulations by agencies like SEC and FINRA, helped curtail adversarial manipulation of execution ordering [17]. Since many blockchain DeFi applications lack strong regulatory bodies, requiring order-fairness from underlying consensus protocols can prevent attacks on transaction ordering.

1.1 Our Contributions and Technical Overview

The main contribution of our paper is to construct fair ordering protocols in the permissionless setting. Towards this end, we introduce the first formalization of a fair transaction ordering property in the permissionless setting by generalizing *order-fairness* from KZGJ. We then analyze the permissioned Aequitas fair ordering protocol from KZGJ to extract out its key technical pieces to be used in our protocols. We show two different constructions that modify any longest-chain based protocol and endow it with the order-fairness property. We characterize the adversarial threshold below which the new protocols achieve order-fairness. Along the way, we uncover insights related to social choice and voting theory that may be of independent interest. Finally, we also show that order-fairness is also useful in reducing the transaction confirmation latency. Specifically, we show that any protocol that satisfies order-fairness also achieves the coveted zero-block confirmation property, where honest transactions can be confirmed locally even before a single block with that transaction has been mined. We briefly elaborate on each of these technical contributions below.

Defining order-fairness in the permissionless setting. We start with the order-fairness property from KZGJ. In the permissioned setting, where the n system nodes were fixed, order-fairness was defined in terms of being received by a large fraction (parameterized by γ) of nodes. For the permissionless setting however, since new nodes may join the network, it does not make sense to consider a fraction of all nodes that ever existed in the system. Furthermore, it is not relevant to consider the transaction input ordering of a node that spawned much later either. Instead, to determine the ordering of a transaction, we will look at the system at the time when the transaction was initially propagated through the network. Intuitively, we introduce the notion of receive-order-fairness (resp. block-order-fairness) in the permissionless setting as the property that tx will be ordered before (resp. no later than) tx' if the initial propagation of tx through the network was earlier than tx'. Following KZGJ, we assume that time is discretized into rounds and transactions that arrive in the same round receive the same timestamp. Thus, the duration of a round defines the coarseness at which we define fair ordering. For example, setting the network delay to $\Delta = 1$ corresponds to looking at fair ordering at the same resolution as network latency. We formally define order-fairness in Definition 2.3.

We note that permissionless order-fairness is not particularly interesting for completely dynamic adversaries. For instance, such an adversary can quickly corrupt and kill all the nodes that were online during the initial propagation of a transaction, before they have a chance to mine any blocks. Any new honest nodes that spawn later will not retain the transaction ordering received by earlier nodes. Therefore, we will restrict our adversaries to be (τ, R)-respawning, i.e., in any R round

period, at most τ fraction of nodes can be killed (and respawned). Note that we still allow the adversary to corrupt nodes instantly.

We also modify the network formalism from KZGJ. KZGJ considered two distinct networks, an *external* network for communication between users and system nodes, and an *internal* network for communication amongst system nodes. While this is not strictly necessary, it prevents an adversarial node that receives a user transaction first, from inserting its own transaction and propagating that through the network earlier (see Section 2.1 for details). The distinction between the external and internal network is less realistic in the permissionless setting where nodes use a gossip-style communication channel, and therefore, we model only a single network. We discuss our modeling choices in depth in Section 2.

We consider a few other possible definitions for fair ordering (Section 2.3). We choose to extend the definition from KZGJ since it is stronger than other notions and better suited for dealing with order-manipulation attacks. Furthermore, receive-order-fairness generalizes the validity condition of the related Byzantine agreement problem (see KZGJ), providing a useful theoretical motivation.

Distilling the Aequitas protocol (from KZGJ) and its connections to voting theory. The Aequitas protocol was constructed by KZGJ for realizing order-fairness in the permissioned setting. At a high level, Aequitas builds a (directed) graph whose vertices contain transactions and edges represent ordering dependencies. A transaction tx can be removed from the graph and output to the final log, when it is in a source vertex, i.e., its ordering does not depend other transactions in the graph. In this paper, we extract out the key techniques from the Aequitas protocol so that they can be adapted for our permissionless protocols through a "player replaceability" lemma (Lemma 3.8), which helps handle nodes dynamically joining and leaving. This lemma will effectively allow us to reduce the permissionless order-fairness problem to its permissioned analogue by creating virtual nodes. Along the way, we find fundamental connections to social choice and voting theory. The problem of choosing the winner of an election (or more generally, a ranking of candidates) based on voter preferences is closely related to the problem of order-fairness. Intuitively, ordering one transaction before another is analogous to preferring one candidate over another. A simple connection in this direction was observed by KZGJ, but we expand on this connection further in Section 3. Specifically, to suit our purpose, we formulate a streaming fair order problem where new transactions (candidates in the voting context) can be added in a continuous fashion. We show how an algorithm that solves this problem can be used to build a fair ordering protocol in the permissionless setting.

Permissionless protocols for order-fairness. Starting from the pioneering idea in [32], longest-chain based protocols have been constructued for many different permissionless settings: proof-of-work [22, 33], proof-of-stake [14, 16, 27] and proof-of-space [38]. We provide two different constructions that modify any longest-chain based protocol and endow it with the order-fairness property. While our constructions are generic to any longest-chain protocol, we describe them in the context of proof-of-work (PoW) for concreteness. Both our constructions follow the same basic outline. First, through PoW mining, nodes mine blocks, and from the blockchain, one can deduce k distinct transaction-order lists, represented by $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_k]$. We refer to these lists as different "semantic chains" since transactions in the same list are logically or semantically connected even though they may directly follow a different PoW block. All transactions will be mined k times, once in each semantic chain. Effectively, $\mathcal{L}$ will represent the transaction orderings

of k virtual "nodes" in a permissioned setting. This allows any node to use the distilled version of the Aequitas protocol (with the player-replaceability property), which we denote by $\mathsf{Aequitas}(\cdot)$, to extract the final fair ordering. For both these protocolos, we characterize the adversarial threshold below which they achieve order-fairness in the PoW setting. Since our proofs are built on basic chain-quality and growth properties of the underlying blockchain, the proofs can be extended to the other permissionless settings such as proof-of-stake and proof-of-space as well. We briefly describe our constructions below.

- (Single-Chain protocol Π_{mod}: Section 5) Our first protocol uses a single PoW Nakamoto chain. Here, the k semantic chains will arise from the chain *modulo* k. Specifically, blocks at indices that are congruent modulo k will become part of the same semantic or *modular* chain. This allows any node to construct the k lists from the public blockchain and run $\mathsf{Aequitas}(\cdot)$ to retrieve the final fair ordering. Assuming (τ, R)-respawning adversaries that corrupt at most β fraction of nodes, for a sufficiently large R, we show that Π_{mod} satisfies order-fairness when $\beta + \tau < \frac{1}{3}$. We comment that using an underlying mining-fair chain like Fruitchains [34] instead, we can tolerate $\beta + \tau < \frac{1}{2}$.
- (Multi-Chain protocol Π_{multi}: Section 6) Our second protocol, Π_{multi}, has a more parallel design for the semantic chains, with the k PoW chains being mined simultaneously. It achieves lower latency than our single-chain protocol, and more importantly, it presents a method to combine multiple ledgers in a fair way. For this protocol, we use the k-for-1 PoW mining technique modified from [7, 22, 34] to determine which chain a block gets mined into. Once again, we can now run $\mathsf{Aequitas}(\cdot)$ to retrieve the final fair ordering. Assuming (τ, R)-respawning adversaries that corrupt at most β fraction nodes, we show security when $\beta + \tau < \frac{7-\sqrt{17}}{16} \approx 0.1798$. The proof establishes a novel cross-chain-quality property using techniques from the queuing theory literature and may be of independent interest to the consensus community.

To show order-fairness, we first define dependency-soundness (Property 4.1) as the property that a majority of the semantic chains in $\mathcal{L}$ contain tx before tx$'$ whenever tx is received before tx$'$ by a large fraction of the network. We show in Lemma 3.8 that any protocol that satisfies this property will achieve order-fairness. Finally, we will show that both of our protocols are dependency-sound. For our constructions, it turns out to be sufficient to show that a majority of blocks that contain a given transaction (across the semantic chains) are honestly mined. Our protocols will satisfy liveness when the network delay parameter is $\Delta = 1$. For this, we show in Lemma 4.3, that when $\Delta = 1$, cycles in the dependency graph can only extend for a small (constant) number of rounds. This allows transactions to be delivered without needing to wait for an arbitrarily large amount of time. Abstractly, the condition of $\Delta = 1$ will mean that the *granularity* to which we can achieve fair ordering (with liveness) is exactly the granularity in which transactions can be input.

After the above reductions take effect, the rest of security proofs for Π_{mod} are reasonably straightforward and rely on properties of the underlying Nakamoto PoW chain proved in [33]. The analysis for Π_{multi} is somewhat complicated, primarily due to possibility of adversarial transactions and block withholding, and the difference in the lengths of different semantic chains. Here, we utilize the cross-chain-quality property for which we make novel use of queuing theory techniques. We defer the complete details for this to Appendix A. We see this work as laying the foundation for the study of order-fairness in the permissionless setting.

Order-Fairness implies zero-block confirmation. Our single-chain protocol provides order-fairness, but it comes at the cost of increased latency. While the multi-chain protocol reduces this latency, it still requires a latency at least similar to Bitcoin (which increases linearly with the security parameter [22]). We show a surprising result here that while the order-finalization latency may be larger, for both protocols, and in fact, for any fair-ordering protocol, any honest transaction can be finalized much faster, *within two-times the network latency*. We note that while there is a large body of work trying to improve the confirmation latency of permissionless systems [7, 20, 30, 41], none of them are able to achieve fast confirmation at this time-scale, since all of them require mining at least a certain number of blocks at a much slower rate than network latency.

To formalize this, we define a useful *soft-ordering property*: if a transaction tx is received at a honest node more than 2Δ time before tx', then tx is guaranteed to be ordered before tx' in the finalized ledger. We show that this property is satisfied by our protocols, and more broadly by any protocol with receive-order-fairness (see Section 7). Soft-ordering is a very powerful property that lets a node confidently predict something about the finalized ledger purely using local observations. Suppose an honest node sees tx in round r and does not see a conflicting transaction within 2Δ in its arrival queue, it can be assured that tx will be ordered ahead of any conflicting transaction in the finalized ledger. Consequently, *a node can confirm a transaction without even seeing a single mined block that contains the transaction*. We call this zero-block-confirmation or soft-confirmation. We note that a transaction issued by a honest client will not have a conflicting transaction at all, and thus any other honest node will be able to *soft-confirm* the transaction within 2Δ rounds.

To see why this is useful, consider the following scenario: Alice wishes to buy a coffee from Carol's Coffee, using a cryptocurrency BasicToken (BT), which uses an underlying Nakamoto-style PoW blockchain. Alice pays 5 BT for her coffee and leaves the coffee shop. However, for Alice's transaction (tx) to be confirmed by the network, it could take several PoW blocks to make sure the transaction is buried sufficiently deep in the chain. This could be in the order of hours, which makes BT based payment systems untenable for practical transactions. If BT were instead run with an underlying blockchain that satisfies order-fairness, then Carol could watch the network for a small period 2Δ to ensure that no conflicting transaction was submitted to the network, and *soft-confirm* the transaction locally. The soft-ordering property lets Carol be locally convinced that tx will be valid in the final ordered ledger, even though tx has not yet been mined, and long before the finalized order of tx is established in the network.

Zero-block confirmation has been a topic of interest from the early days of Bitcoin, and touted as an important step towards the practicality of Bitcoin for day-to-day purchases. Practictioners use many ad-hoc techniques [25], based on listening to the network for a small period of time to ensure there was no conflicting double-spend transaction, and gossiping any conflicting transaction to the network. However, since Bitcoin does not guarantee fair-ordering, it is possible for a transaction to be issued much later and still be ordered ahead. A simple method is for a malicious user to issue a second double-spend transaction that gets mined through an adversarial miner. In practice, this can be done by using a higher transaction fee. More long range attacks are also possible through selfish mining [18]. A fair ordering protocol ensures that such a behavior is not possible, and guarantees that the soft-confirmation is indeed provably secure.

Paper Organization. The rest of the paper is organized as follows. We discuss some related work in Section 1.2. We define our formal model and other preliminaries in Section 2. In Section 3, we distill the Aequitas protocol from KZGJ, and connect it to work in voting theory. Section 4

provides an overview of our fair ordering protocols as well as useful theorems that help modularize our proofs. We construct our single-chain protocol in Section 5 and our multi-chain protocol in Section 6. Some of the proofs for our multi-chain protocol are computationally intricate and we defer them to Appendix A.

1.2 Related Work

We describe related works in this section. The state machine replication, or consensus abstraction has been studied in literature for several decades but almost no prior permissioned protocols, and no permissionless protocols consider any notion of fairness in transaction ordering. For a more comprehensive comparison to prior permissioned broadcast and consensus protocols, we refer the reader to the related works section in KZGJ.

Fair ordering protocols. Very little research has been done on fair ordering consensus protocols. Hashgraph [8] provided an informal description of a first-in-first-out (FiFo) transaction ordering property. KZGJ [26] formally defined *(receive)-order-fairness* (resp. (block)-order-fairness) as the property of ordering tx_1 before (resp. no later than) tx_2 globally if a large fraction of nodes (parameterized by γ) received tx_1 before tx_2 in their local ordering. Concurrently to KZGJ, works by Kursawe [28], and Zhang et al. [42] gave an alternative definition for ordering fairness (called *timed relative fairness* in [28] and *fair-linearizability* in [42]) where tx_1 is ordered before tx_2 if all honest nodes have seen tx_1 before *any* honest node has seen tx_2. Note that this is strictly weaker than the receive-order-fairness property from KZGJ and in Section 2.3, we argue why it is not enough to prevent natural order-manipulation attacks. Still, work on fair ordering protocols so far, has been restricted only to the permissioned setting, despite most motivating attacks taking place in the permissionless setting. To the best of our knowledge, our work is the first to construct fair ordering protocols in the permissionless setting.

Alternate ordering techniques. An extensive line of research ([22, 33, 34] among others) in the past few years has considered the consensus problem in the permissionless setting. By design, most permissionless protocols achieve *censorship resistance*, which ensures that any transaction submitted by an honest user will eventually be confirmed by the network. Despite this, no guarantees are provided on how the transaction is eventually ordered.

Some techniques like [9, 31] attempt to prevent adversarial manipulation of the transaction ordering by finalizing the ordering before revealing the transaction contents, or choosing a random ordering among the transactions in the current pool. Even in such a protocol, an adversary can introduce new transactions to the network such that with a high probability, one of them will get sequenced before an honest transaction even though the honest one was received earlier [3]. This is problematic if the adversary can use user transaction metadata to construct its own transaction, or if the adversarial transaction does not need to depend on the contents of any honest user's transaction (e.g., the adversary wants to be the first one through the door to get the best price in an auction). Therefore, a random ordering protocol is not sufficient to prevent order-manipulation attacks. We elaborate on this observation in Section 2.3.

Other unrelated uses of fairness. The term *fairness* has been used previously in literature for properties unrelated to ours. In permissionless consensus literature, one common use case is

for PoW block mining; it requires that an honest node's rewards be proportional to its relative computational power, which restricts adversarial *selfish mining* [18]. This notion of fairness was first defined by Pass and Shi [34] and achieved by their Fruitchains protocol. We will call this property *mining-fairness* to distinguish it from our notion of order-fairness.

Latency reduction methods. Reducing the confirmation latency of longest-chain protocols has been the subject of intensive recent research [7, 20, 30, 41]. Two particularly relevant works, Prism [7] and Ledger-Combiner [20], also use multiple parallel chains to achieve faster confirmation. They can confirm honest transactions within a constant multiple of the network latency Δ, much faster than Bitcoin, where the latency is a multiple of the security parameter. However, in both methods, the constants are large and become worse as the adversarial power increases. Other methods [30, 37] offer low latency only under optimistic conditions, and fall back to standard Bitcoin latency when there are any adversarial actions. In comparison, our soft-confirmation method can confirm honest transactions in 2Δ time independent of the tolerable adversary ratio or whether there is an active attack. Conceptually, our soft-confirmation method can confirm transactions that are yet to get into any block, which is impossible with the other methods. Surprisingly, Ledger-Combiner, which was specifically designed to get low latency when combining transactions from multiple ledgers is slower than the soft-confirmation our constructions achieve as a byproduct.

2 Model and Preliminaries

We describe the general framework for protocol execution in this section. Much of our formalism is adapted from an extensive line of research on consensus in the permissionless setting [22, 33, 34, 36, 37]. We also use some of the formalism related to order-fairness from KZGJ [26]. Lifting the network and order-fairness formalism from KZGJ to the permissionless setting brings out several subtleties which we detail in Sections 2.1 and 2.3.

General Execution Model. To model protocol execution, we adopt the widely used Interactive Turing Machine (ITM) approach from the Universal Composability framework [10]. Each node is represented as an ITM and a special environment machine $\mathcal{Z}$ is used to direct the protocol's execution. We assume the existence of an underlying global clock that allows us to model protocol execution in *rounds*[2]. Specifically, at the start of each round, each node receives inputs from $\mathcal{Z}$ in the form of a set of transactions txs; at the end of every round, nodes may deliver[3] outputs to $\mathcal{Z}$. For ease of modeling, for honest nodes, the final ledger will be taken as the result of a transformation function $\mathsf{linearize}(\cdot)$ (defined as part of the specific protocol) on the outputs to $\mathcal{Z}$ rather than the outputs themselves. That is, for an honest node i in any round t, the final ledger is the result of $\mathsf{linearize}$ on i's output to $\mathcal{Z}$ in round t. This will extract the relevant information from the outputs in the form of a linearly ordered ledger. New transactions from users are modeled as being input by $\mathcal{Z}$ to avoid explicitly having to model clients.

Corruptions. At any point, $\mathcal{Z}$ can spawn new nodes, either as *corrupt* or *honest*. Honest nodes execute the protocol faithfully while corrupt nodes are controlled by an adversary $\mathcal{A}$, and can

[2]In this paper, we will use the terms *time* and *rounds* interchangeably.
[3]In this paper, we will use the terms *deliver* and *output* interchangeably.

deviate arbitrarily. At any point, $\mathcal{Z}$ can send a `corrupt` signal to a node (perhaps as requested by $\mathcal{A}$), after which $\mathcal{A}$ gets control of the node's execution. At any point, $\mathcal{Z}$ can also send a `kill` signal to an already corrupt node, which removes it from the protocol execution. Note that only corrupt nodes are killed and that $\mathcal{A}$ knows the internal state of a node right before it is killed. Corrupt nodes can also be uncorrupted by $\mathcal{Z}$ and this is treated the same way as first killing the node and then spawning a new node.

We observe that if the adversary is able to corrupt and kill nodes at will, it can cause the network to "forget" the transaction input orderings received by earlier honest nodes. While this was not important for prior work, where the final transaction ordering was not considered, this is critically detrimental for any fair ordering protocol. Therefore, we will assume a *(mildly)-respawning-adversary* that limits the node churn. Informally, a (τ, R)-respawning[4] adversary can kill at most τ fraction nodes within any period of R rounds. This corresponds to the notion that $\mathcal{A}$ cannot quickly corrupt and kill many nodes. We will distinguish between the respawning parameter and the adversarial corruption parameter (see Definition 2.1).

Notational conventions. We use κ to denote the security parameter. For a protocol Π, $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ represents the random variable for all possible execution traces of Π w.r.t. adversary $\mathcal{A}$ and environment $\mathcal{Z}$. The possible executions arise from any randomness used by honest nodes, adversarially controlled nodes, and the environment. Any view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ is a fully specified instance (including inputs, outputs, random coins etc.) of an execution trace. We use $\mathsf{view} \leftarrow\!\!\$\ \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ to denote randomly sampling an execution. $|\mathsf{view}|$ denotes the number of rounds in view. A function $\mathsf{negl}(\cdot)$ is *negligible* if for every polynomial $p(\cdot)$, there exists a constant $\kappa_0 \in \mathbb{N}$, such that $\mathsf{negl}(\kappa) \leq \frac{1}{p(\kappa)}$ for all $\kappa \geq \kappa_0$.

2.1 Communication Networks

Before we introduce our network formalism, we look at the formalism in KZGJ to better argue for the modeling choices we make for the permissionless setting.

A deep look into the networks in KZGJ**.** Standard formalism for consensus protocols models a single communication network amongst consensus nodes. Informally, network synchrony here implies that any message sent by an honest node reaches its intended recipient(s) within some known fixed amount of time (or rounds) Δ. Usually transactions from clients are assumed to originate either from a consensus node or input by the environment $\mathcal{Z}$.

The order-fairness formalism from KZGJ, however, models two separate networks: an external network (which models the communication channel between system users and protocol nodes), and the internal network (which models the communication network amongst protocol nodes and is the standard one considered in literature). Furthermore, the model assumes that transactions are sent by clients to all nodes (modeled as inputs given by $\mathcal{Z}$), and that the adversary only controls network delivery for the internal network. Synchrony in the external network is defined as the property that any transaction input to one node (by $\mathcal{Z}$) is input to all other modes within some

[4]This is distinguished from an R-mild adversary in [35] where the adversary takes R rounds to *corrupt* a node. While the security properties of [35] crucially relied on the adversary not gaining control of nodes instantly, we can tolerate the adversary corrupting a node instantly, as long as it can't kill too many nodes quickly.

known time Δ_{ext}. In practice, this is the assumption that a client's connection to any particular node is not much slower than its connection to others.

Such a modeling was crucial for their purpose because each consensus node needed to have an unmolested view of its transaction input ordering. For example, the first node that receives a client transaction should not be able to adversarially insert its own transaction and quickly gossip it to other nodes in an attempt to get it sequenced first. This is a form of "front-running" as described in [15]. The assumption that the adversary does not control the external network takes care of this problem. However, this puts the onus on the client to send its transaction to all nodes instead of just a few. While this is a perfectly reasonable assumption in a permissioned network where the nodes are known, it is difficult to justify it in a permissionless network, where nodes change over time and transactions usually propagate through a gossip-style network. Here, arguably there is no way to get around nodes inserting their own transactions during the gossip. Nevertheless, we emphasize that this still retains the notion that a transaction that is seen by a majority of the network first, should be sequenced first. We note that the permissioned formalism in KZGJ could very well have modeled a single network, but the additional modeling of the external network allowed an honest node's input ordering to be completely independent of the adversary. In practice, in a permissionless network, clients should send their transactions to as many nodes as they can so that most nodes receive them in an order not influenced by the adversary.

Another upshot of modeling two separate networks in KZGJ was that a transaction claimed by an adversarial node would not be considered by an honest node unless the transaction was received from the environment i.e., an honest client. This works well since now, an adversary cannot influence the global state by creating bogus transactions. Unfortunately, in a permissionless Proof-of-Work protocol, transactions can also be gossiped by mining a block that contains it and propagating the block through the network. It is difficult to determine the final ordering simply based on the local input orderings of all the nodes from clients (like was done in KZGJ) since participating nodes may constantly change in a permissionless system. On the other hand, if transaction input orderings are based on the ordering in the ledger, it could potentially enable an adversarial transaction (that was not input to any honest node), to be sequenced earlier simply by being placed earlier in the ledger. To get around this, informally, we force the environment to deliver transactions included in the ledger to all nodes *as input* which handles transaction discovery through a gossip-network, and allows us to talk meaningfully about transaction ordering.

All of this ends up blurring the distinction between the internal and external network from KZGJ, leading us to model a single communication network.

Network Formalism. We model a single communication network that in spirit melds the external and internal network from KZGJ. We assume a synchronous model of communication, such that messages sent are delivered within a (known) bounded number of rounds Δ. The adversary $\mathcal{A}$ can delay and reorder messages (subject to the bound Δ) but cannot drop messages. We define the synchrony assumption below.

- (Δ-(*bounded*)-*delay*) We say that $(\mathcal{A}, \mathcal{Z})$ respects Δ-delay if the following conditions hold for all view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$:

  - (Input Synchrony) If an honest node receives tx as input from $\mathcal{Z}$ in round r, then in any round $r' \geq r + \Delta$, any honest node that is in the system in round r' will have already

received tx as input from $\mathcal{Z}$. This includes any node that may have spawned in round r'.

- (Ledger Synchrony) If an honest node outputs tx to $\mathcal{Z}$ in round r (recall that this is not the same as being ordered in the final ledger), then in any round $r' \geq r + \Delta$, any honest node that is in the system in round r' will have already received tx as input from $\mathcal{Z}$. This includes any node that may have spawned in round r'.
- (Broadcast Synchrony) If an honest node sends a message in round r, then in any round $r' \geq r + \Delta$, any honest recipient node will have received the message by round $r + \Delta$. This includes any node that may have spawned in round r'.
- $\mathcal{Z}$ provides the delay bound Δ to all nodes when they are spawned.

Since we model a single network, we consider the same synchrony bound for all properties. Although, it is possible to model different synchrony parameters, in practice, it is unlikely that they will be significantly different.

Remark (Input Synchrony and DoS). Similar to the external synchrony assumption in KZGJ, our input synchrony assumption assumes that all transactions are gossiped through the network. In a permissionless network, this can lead to a possible denial-of-service (DoS) attack where an adversary floods the network with bogus double-spend transactions (e.g., transactions that spend the same tokens). This is mitigated in the Bitcoin network, by not gossiping any double-spend transactions. Unfortunately, this can cause our zero-block confirmation property to be invalidated. Instead, we will show that gossiping a single double-spend transaction is sufficient to retain our zero-block confirmation property. In other words, the adversary cannot cause a DoS attack by attempting to flood the network. We elaborate on this further in Section 7. We also note further that the proposal in [25] to prevent fast double-spends (although as mentioned earlier, it cannot guarantee that later double-spends are not sequenced earlier due to adversarial miners) involves gossiping all double-spends through the entire network, for which the authors argue that the performance penalty incurred by the network would not be significant.

2.2 Abstract Blockchain Protocols and Formalism

We recall the definition of the permissionless or open setting next, taken from [37], and add to it our respawning parameters.

Definition 2.1 ($(n, \beta, \Delta, \tau, R)$-permissionless environments)**.** We say that $(\mathcal{A}, \mathcal{Z})$ respects $(n, \beta, \Delta, \tau, R)$-permissionless execution w.r.t. a protocol Π if $(\mathcal{A}, \mathcal{Z})$ respects Δ-bounded-delay, and for every $\kappa \in \mathbb{N}$ and view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the following conditions hold:

- In each round, there are exactly n nodes online in the system, of which at most βn are corrupt.
- Within any period of R rounds, at most τn notes are killed. We call such an adversary (τ, R)-respawning.
- $\mathcal{Z}$ provides all nodes the parameters $n, \beta, \Delta, \tau, R$ upon spawning.

Remark. (Variable n) While we model a fixed n in each round similar to most blockchain papers, we note that our model can easily be extended to handle a varying n, using complementary techniques that are well understood in literature [11, 21]. This involves adjusting the mining difficulty target and using a heaviest (i.e., total amount of work) chain rule rather than the usual longest chain rule.

Next, we briefly recall the formalism for abstract blockchain protocols. An abstract blockchain chain is an ordered sequence of blocks where $\mathsf{chain}[i]$ is a vector of transactions contained in chain at index i. We assume familiarity with the (by now) standard requirements for blockchain protocols, and refer the reader to [22, 33, 37] for more details. Our proofs will require these properties from previous well-studied constructions only in a black-box way.

Notation. For a given chain and non-negative integers d and d', we use $\mathsf{chain}[d]$ to denote the d^{th} block of chain (indexed from 0), $\mathsf{chain}[:d]$ to denote the first d blocks, and $\mathsf{chain}[d:d']$ to denote the sequence of blocks from $\mathsf{chain}[d]$ to $\mathsf{chain}[d']$, both inclusive. We also use $\mathsf{chain}[-d]$ $(d \neq 0)$ to denote the d^{th} block from the rear and $\mathsf{chain}[:-d]$ to denote chain except for the last d blocks.

Definition 2.2 (Abstract Blockchain Requirements)**.** The requirements below need to be satisfied, except with negligible probability, over a random choice of view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$.

- (T-Common-Prefix) If chain^r and chain^t are two honest chains (possibly from the same node) in rounds r and t respectively, and $r \leq t$, then $\mathsf{chain}^r[:-T] \preceq \mathsf{chain}^t$. This will allow us to confirm all except the last T blocks in chain.
- ((T, g_0, g_1)-Chain-Growth) Let chain^r and chain^t be two honest chains (possibly from the same node) in rounds r and t respectively such that $r \leq t$.
  - (Consistent length) If $t \geq r + \Delta$, $\left|\mathsf{chain}^t\right| \geq |\mathsf{chain}^r|$.
  - (Chain growth lowerbound) If $g_0 \cdot (t - r) \geq T$, then $\left|\mathsf{chain}^t\right| - |\mathsf{chain}^r| \geq \lfloor g_0 \cdot (t - r) \rfloor$.
  - (Chain growth upperbound) If $g_1 \cdot (t - r) \geq T$, then $\left|\mathsf{chain}^t\right| - |\mathsf{chain}^r| \leq \lceil g_1 \cdot (t - r) \rceil$.
- ((T, μ)-Chain-Quality) If chain is an honest chain, then for any T consecutive blocks in chain, at least μT blocks were mined by honest nodes.

Protocols that satisfy the blockchain abstraction will directly satisfy the state machine replication properties of consistency and liveness (see [33, 37]). We use the following definition for liveness.

- ($(T_{\text{warmup}}, T_{\text{confirm}})$-liveness) If an honest node is input m by $\mathcal{Z}$ in round $r \geq T_{\text{warmup}}$, then in any round $r' \geq r + T_{\text{confirm}}$, for any honest node in round r' that outputs $\mathcal{X}$ to $\mathcal{Z}$, m is contained in $\mathsf{linearize}(\mathcal{X})$. When there is no warmup time, we simply write T_{confirm}-liveness.

We introduce useful formalism for Nakomoto's PoW protocol next.

Random Oracle. We assume that all nodes have access to a random function $H : \{0,1\}^* \to \{0,1\}^\kappa$. This is provided by two oracles, $\mathsf{H}(x)$ which outputs $H(x)$ and $\mathsf{H.ver}(x, y)$ which outputs 1 if $H(x) = y$ and 0 otherwise. In any round, all parties can make q queries to H and any number of queries to $\mathsf{H.ver}$. An adversary $\mathcal{A}$ controlling p parties is allowed to make pq sequential oracle queries. Note that $\mathcal{Z}$ cannot access H, but can instruct $\mathcal{A}$ to make queries. This abstraction follows [22, 33], and captures the intuition that finding a proof-of-work solution is difficult, while checking the validity of a solution is inexpensive.

Blocks. We define a block $\mathtt{B}$ as a tuple $(\mathtt{parent}, \mathtt{ref}, \mathtt{m}, \eta, h) = (h_{-1}, h', \mathtt{m}, \eta, h)$, where $\mathtt{parent}$ denotes a pointer to a parent block, $\mathtt{ref}$ denotes a pointer to a reference block (an arbitrary previous block), $\mathtt{m}$ denotes the record in the block (eg., a vector of transactions), η denotes the proof-of-work nonce, and h denotes the hash of the current block.

The reference block is an important modeling choice since it allows $\mathtt{B}$ to *logically* follow a different block than its preceding parent block. Notably, this allows the validity condition for a block to also be defined in terms of the reference block, and possibly separately from the parent block. We note that in many cases (eg., the Bitcoin backbone protocols), $\mathtt{ref} = \mathtt{parent}$. In such protocols, we can condense the size of the block by dropping any unnecessary values or replacing them with a hash digest to allow for verification.

We will parameterize our protocols by a hardness function $p(\cdot)$ that defines $D_p = p(\kappa) \cdot 2^\kappa$ such that $\Pr_\eta[H(\mathtt{parent}, \mathtt{ref}, \mathtt{m}, \eta) < D_p] = p(\kappa)$ for all $(\mathtt{parent}, \mathtt{ref}, \mathtt{m})$. Now, we define a valid block $\mathtt{B}$ as a tuple $(h_{-1}, h', \mathtt{m}, \eta, h)$ where $H(h_{-1}, h', \mathtt{m}, \eta) = h$ and $h < D_p$.

Security of Nakamoto's protocol [33]. Consider Nakamoto's protocol $\Pi_{\text{nak}}(p)$ with hardness parameter $p(\cdot)$. Let α be the probability that some honest nodes mines a block in one round and let ν be the expected number of adversarial blocks mined in one round. Different quantities are considered for honest and adversarial nodes since any honest chain can grow by at most one block in any round while an adversary, who can query the H sequentially can append more than one block. Then, $\alpha = 1 - (1 - p(\kappa))^{(1-\beta)qn}$ and $\nu = \beta qnp(\kappa)$ [33][5]. Define $\alpha' = \frac{\alpha}{1+\alpha\Delta}$. Define a Nakamoto-compliance predicate, denoted by $\Gamma^p_{\text{nak}}(n, \beta, \Delta) = 1$ if for all κ, there is some $\lambda > 1$ such that $\alpha(1 - 2(\Delta + 1)\alpha) \geq \lambda\nu$. Then, in any environment that satisfies the Nakamoto-compliance predicate, or equivalently, for a sufficiently small hardness parameter, any $T_0 = \omega(\log \kappa)$, and any constant $\varepsilon > 0$, $\Pi_{\text{nak}}(p)$ satisfies T_0-consistency; (T_0, g_0, g_1)-chain-growth where $g_0 = (1 - \varepsilon)\alpha'$, $g_1 = (1 + \varepsilon)qnp$; and (T_0, μ) chain quality where $\mu = 1 - (1 + \varepsilon)\frac{\nu}{\alpha'}$. Note that all these quantities are functions of κ.

Proof simplifications. To make our proofs cleaner, we will make the following standard simplifications.

- (Simplification 1) (Mining is much slower than block propagation.)

  When the hardness parameter is sufficiently small, (e.g., when pq is small compared to $1/n$), we have $\alpha \approx q(1-\beta)np$. Therefore, by the union bound, the probability that no honest block is mined in Δ rounds is at most $(\Delta)\alpha$. If this is small, then with overwhelming probability, no other honest blocks are mined in the time it takes for an honestly mined block to be propagated to other nodes. This is reasonable since network propagation in Bitcoin is a few seconds, while a new block is mined roughly once every 10 minutes. For sufficiently small hardness parameter $p(\cdot)$, Pass et al. [33] showed that this happens with overwhelming probability in their proofs. We note that some prior works (eg., Prism [7]) also directly make a similar assumption.

- (Simplification 2) When the hardness parameter is sufficiently small, $\frac{\nu}{\alpha'}$ is well approximated by $\frac{\beta}{1-\beta}$, and therefore $\mu = 1 - (1 + \varepsilon)\frac{\nu}{\alpha'}$ is well approximated by $1 - (1 + \varepsilon)\frac{\beta}{1-\beta}$. Then,

[5] We consider q queries to H per round for each node whereas [33] considers a single query.

for any $\beta < \frac{1}{3}$, there will exist some ε_0 such that Π_{nak} satisfies chain quality with $\mu > \frac{1}{2}$. Specifically, for any $\epsilon > 0$ and $\beta = 1/3 - \epsilon$, let $0 < \varepsilon_0 < 1 - \frac{2+3\epsilon}{2-\epsilon}$. Then, Π_{nak} satisfies (T_0, μ_0)-chain quality where $T_0 = \omega(\log \kappa)$ and $\mu_0 > \frac{1}{2}$. In other words, in any T_0 consecutive blocks, the majority of blocks are mined by honest miners. In our proofs, we will directly use majority-chain-quality when $\beta < \frac{1}{3}$.

2.3 Order-Fairness

Background on order-fairness. An extensive number of transaction reordering and front-running attacks [3, 15, 17], have been observed recently on real-world blockchains; in particular, in the context of Decentralized Apps (or DApps) on Ethereum. While some prior protocols have attempted to provide partial fixes for ordering attacks, KZGJ initiated the first formal study for fairness of transaction ordering in the permissioned setting. Informally, KZGJ defines γ-receive-order-fairness (resp. γ-block-order-fairness) as the following property: If γ fraction of nodes receive as input a transaction tx before another one tx$'$, then tx should be included before (resp. no later than) tx$'$ in the final ordering. We emphasize (as shown in KZGJ) that receive-order-fairness is a strictly stronger ordering property than previously considered notions like censorship resistance, random leader election, threshold encryption etc.

While KZGJ studied fair transaction ordering in the permissioned setting, since almost all of the aforementioned real-world attacks are on permissionless systems, it is important to extend the order-fairness definition to the permissionless setting. We will do exactly that in this section.
Before proceeding with our formalism, we briefly mention two alternate candidate definitions for fair ordering, that are weaker than receive-order-fairness (and perhaps easier to achieve), but cannot prevent order-manipulation through front-running.

- (Timed relative fairness [28] or fair-linearizability[6] [42] from works concurrent to KZGJ) If all honest nodes receive tx before any honest node receives tx$'$, then tx will be ordered before tx$'$. In other words, if the latest honest receive time for tx is before the earliest honest receive time for tx$'$, then tx should be ordered earlier.

  Note that this is strictly weaker than receive-order-fairness. Indeed, if the antecedent of fair-linearizability is satisfied, then so is the antecedent of receive-order-fairness. The primary goal of the fair-linearizability property was to add an ordering property to existing consensus protocols, and indeed, we find that this is not sufficient to handle our motivation of front-running attacks. This is because it enforces no guarantees on transactions whose receive times are globally interleaved. An adversarial transaction that attempts to front-run an honest user transaction, will end up getting interleaved with the user transaction in its receive times. Consequently, for most practical attack vectors, the antecedent of fair-linearizability will be false, making the definition vacuous and not particularly useful. We also note that this definition lacks a parameterization by γ, compared to the order-fairness definition.

  We do show however, that this property is sufficient for zero-block confirmation.

- (Median-Fairness) If the median receive time for tx is smaller than the median receive time for tx$'$, then tx should be ordered before tx$'$.

[6] Fair-linearizability allows the outcome where only tx$'$ is ordered and tx never is, effectively allowing some form of censorship. It is possible to prevent this but comes at the cost of increased latency.

KZGJ considered this definition, but showed that it deviated from the intuition of "first-in first-out" ordering since a single adversarial node could flip the median, thereby resulting in an unfair ordering.

Random ordering vs Fair ordering. It is important to distinguish a protocol that chooses a *random* ordering from a protocol that chooses a *fair* ordering. A *random* ordering protocol chooses a random ordering of transactions in the current pool, in an attempt to avoid any adversarial manipulation. On the other hand, a fair ordering protocol will guarantee that transactions which arrive earlier will be sequenced earlier by the network (instead of in a random order). Note that in a random ordering protocol, there is still a 50% chance that an adversarial transaction gets ordered before an honest one even if the honest one was propagated earlier. In addition, the adversary can flood the network with many transactions to exponentially increase the probability that at least one is ordered before the honest transaction. We highlight that even if the adversary has no control over the ordering of transactions in the pool, it is able to implicitly influence the ordering by introducing its own adversarial transactions.

Some protocols [6, 9, 31] make the use of threshold encryption, or a commit-and-reveal scheme, to order transactions before their content is revealed in order to choose a random ordering. In practice, this might still not prevent censorship and reordering based on transaction metadata (user identifier, client IP address etc). Furthermore, it may not prevent front-running based on information received through side channels. Still, even if these problems were mitigated, we maintain that random ordering is not the same as fair ordering. We also note that our techniques can easily integrate with commit-and-reveal methods where the final random ordering is replaced with the fair ordering based on transaction receive times.

Choosing an order-fairness definition in the permissionless setting. Consider two transactions tx and tx'. First, we note that in the permissionless setting, nodes that are spawned far later (eg., after tx and tx' are output), will receive tx and tx' in the same round (see the input synchrony definition). Consequently, we cannot naïvely use the same order-fairness definition from the permissioned setting, since the antecedent will never be satisfied making the definition vacuous. Instead, we only consider the first time that the n nodes in the system receive either transaction, corresponding to the initial propagation of the transactions.

Let $\mathcal{S}$ be the set of these nodes. We say that a node has received tx before tx' if it has either (1) received tx but not tx' or (2) received both transactions and received tx in an earlier round than tx'. Now, if a large fraction (parameterized by γ) of nodes in $\mathcal{S}$ have received tx before tx', then intuitively, tx should be preferred over tx'. For receive-order-fairness (resp. block-order-fairness), this means that the final log should not contain tx' unless it contains tx before (resp. in the same block). Henceforth, for brevity, when we say that the final log contains tx before tx', we also include the scenario where the log only contains tx. Permissionless order-fairness is formalized in Definition 2.3.

Definition 2.3 (γ-receive-order-fairness (resp. γ-block-order-fairness))**.** For a given view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, we define γ-receive-order-fairness (resp. γ-block-order-fairness) as follows: For two transactions tx and tx', let r be the first round such that at least n nodes alive in round r, have received either tx or tx'. Now, if γ fraction of those nodes have received tx before tx' as input, then for any round $t \geq r$, and any node i that is honest in round t and outputs $\mathcal{X}$, the

final linearly ordered log, $\mathsf{linearize}(\mathcal{X})$, either (1) contains neither transaction; (2) contains only tx; or (3) contains tx before (resp. no later than) tx$'$.

We say that a protocol satisfies order-fairness if it satisfies the above, except with negligible probability over a random choice of view.

3 Distilling the Aequitas approach

We distill the key structure of the Aequitas protocol, as described in KZGJ, that provides order-fairness in the permissioned setting. We generalize their techniques to be useful in the permissionless setting we consider. Along the way, we uncover some helpful insights and connections to voting systems that may be of independent interest.

First, we briefly describe the key techniques employed by the Aequitas protocol. While the protocol works for both synchronous and asynchronous settings, we will only consider the *completely synchronous* setting where both the internal and external network are synchronous. In the permissioned setting, consider a system with n nodes, of which, at most f may be corrupt (controlled by $\mathcal{A}$). The protocol is parameterized by an order-fairness parameter $\frac{1}{2} < \gamma \leq 1$ that dictates what constitutes a fair ordering. In the synchronous setting, for parameter γ, the Aequitas protocol requires $n > \frac{2f}{2\gamma-1}$. At the start of a round, each node receives transactions as input from $\mathcal{Z}$. For a node i, let $\mathsf{TxInputOrder}_i$ denote the ordered list of transactions as received by i. The first stage of the Aequitas protocols uses broadcast primitives (specifically FIFO (first-in-first-out) broadcast [24]) to let all nodes know the order of transactions as received by other nodes. While adversarial nodes can choose to broadcast any order as though they received it, FIFO-broadcast along with an intermediate agreement stage ensures that all nodes still have a consistent view of this ordering. The final stage of the Aequitas protocol, dubbed the "finalization stage", requires no further communication and uses the input orderings of all the nodes to locally compute the final output transaction ordering. Each transaction goes through the following three stages before being delivered as part of the final ledger.

1. (Broadcast) In this stage, nodes broadcast their input transaction ordering as received from the environment $\mathcal{Z}$.

2. (Agreement) In this stage, nodes agree on whose input orderings to use to order the given transaction. This is done to ensure that adversarial nodes cannot cause nodes to use a different set of orderings by broadcasting late. In the synchronous setting, this also guarantees that the input orderings of all honest nodes are considered.

3. (Finalization) In this stage, all nodes are now equipped with all the input orderings. At this point, nodes compute the final fair ordering locally.

Essentially, this means that we can abstract out the broadcast and agreement primitives and assume that the nodes start with the transaction input orderings of all nodes; of which f orderings may be adversarial chosen. The "finalization" step now takes these as input to locally compute the final output ordering. Since f of the orderings could be adversarial, for any two transactions tx and tx$'$, the finalization for a protocol with order-fairness parameter γ, must output tx before (no later than in the case of block-order-fairness) tx$'$ even when tx is ordered before tx$'$ in $\gamma n - f$ orderings. The upshot of this observation is it allows us to distill the computation of the fair ordering from the consensus part, and modify it to suit our purpose.

3.1 The Finalization Problem

Before we formally define the finalization problem, we introduce some notation. For our finalization problem parameterized by $\Upsilon = (n, f, \gamma)$, we will consider as input, n ordered lists List_1 to List_n where an entry in a list is a set of messages. While the inputs to the finalization problem are independent of any underlying consensus problem, in the permissioned setting, intuitively, these lists describe the order in which messages were input to a particular node. We use $\mathsf{List}_i[r]$ to denote the the set of messages received by node i in round r (if i is honest). Lists corresponding to adversarial nodes, can of course deviate arbitrarily from their true input ordering. For messages m and m', we use the notation $m \prec_i m'$ if m is ordered before m' in List_i. The goal now, is to output a list $\mathsf{OutputList}$ of transactions that are ordered in a "fair" manner. We say that $\Upsilon = (n, f, \gamma)$ is admissible if $\gamma > \frac{1}{2}$ and $n > \frac{2f}{2\gamma - 1}$. We define a problem instance by $\Lambda = (\Upsilon, \mathcal{L})$ where $\Upsilon = (\Upsilon.n, \Upsilon.f, \Upsilon.\gamma)$ and $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_n]$ indexed by 1 to n.

Definition 3.1 (Valid Message)**.** We say that a message m is valid w.r.t. $\Lambda = (\Upsilon, \mathcal{L})$ if it is present in at least $\Upsilon.n - \Upsilon.f$ lists in $\mathcal{L}$.

Definition 3.2 (γ-Global Preference)**.** Given $\Lambda = (\Upsilon, \mathcal{L})$, for two messages m and m', we say that m is *globally preferred* over m', denoted by $m' \lhd^{\Lambda} m$, if there are at least $(\Upsilon.\gamma)(\Upsilon.n) - \Upsilon.f$ input lists List_j in $\mathcal{L}$ such that $m \prec_j m'$. We will write $m' \lhd m$ when Λ is clear from context.

We also define a Condorcet cycle for non-transitivity in the global preference, which can arise from the Condorcet paradox. See Section 3.1.1 for more details.

Definition 3.3 (Condorcet Cycle)**.** Given $\Lambda = (\Upsilon, \mathcal{L})$, we say that $(m_1, m_2, \ldots, m_l)$ is an l-length Condorcet cycle in $\mathcal{L}$ if $m_1 \lhd^{\Lambda} m_l$ and $m_{i+1} \lhd^{\Lambda} m_i$ for all $i \in \{1, \ldots, l\}$.

We now define the basic static finalization problem. This is intended as a warm-up to understand the finalization problem better and serves as a strawman example for classic techniques from social choice theory.

Problem 3.4. (Strong (resp. Weak) Static Fair-Order Finalization) Consider an admissible $\Lambda = (\Upsilon, \mathcal{L})$ as input. Output a list $\mathsf{OutputList}$ containing all valid messages in $\mathcal{L}$ such that m is ordered before (resp. no later than) m' in $\mathsf{OutputList}$ if m is globally preferred over m' w.r.t. Λ, or equivalently, $m' \lhd^{\Lambda} m$.

3.1.1 Connections to voting theory

It is easy to draw parallels between the Static Fair-Order Finalization problem (Problem 3.4) and majority voting systems, in which voters rank a set of candidates. In such a voting system, a candidate A is preferred over a candidate B if a majority of voters prefer A over B. The goal of a voting protocol now is to find the winning candidate(s), or more generally rank all candidates according to voter preferences. For our purpose, we are essentially replacing candidates with transactions and voters with consensus nodes. We note that Problem 3.4 also generalizes the condition when one transaction is preferred over another. Instead of a simple majority, we parameterize preference using a order-fairness parameter γ (see Definition 3.2). We also allow some of the "voting preferences" to be adversarial, which is not something usually considered in voting theory.

Static Finalization Algorithm **StaticFinalize**

Input: $\Lambda = (\Upsilon, \mathcal{L})$ where $\Upsilon = (n, f, \gamma)$ is admissible and $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_n]$.

Notation: Let $\mathcal{V}$ be the set of valid messages in the input lists.

Algorithm Description.

1. Construct an empty graph $G = (V, E)$, where V denotes the set of vertices, and E denotes the set of edges. For all $m \in \mathcal{V}$, add a vertex labeled m to G.
2. For all $m, m' \in \mathcal{V}$, if $m' \lhd^{\Lambda} m$, then add the edge (m, m') to G.
3. Let G_{con} be the condensation graph of G. Output the topological sorting of G_{con}.

Figure 1: Static Finalization Algorithm

Condorcet cycles and Smith sets. One standard choice for selecting the winner(s) in a majority voting election is the Smith criterion [40]. The Smith set is the smallest possible set of candidates that are preferred over other candidates not in the set. When a Condorcet winner exists (i.e., there are no Condorcet cycles), it is always the sole member of the Smith set. By a Condorcet cycle, we mean a sequence of candidates (or messages), where the global preference relation is non-transitive. Such cycles can exist even when individual orderings are transitive due to the Condorcet paradox [12, 26].

Definition 3.5 ((Majority) Smith Set)**.** The Smith set is the smallest set $\mathcal{S}$ that satisfies the condition that for all $x \in \mathcal{S}$ and $y \notin \mathcal{S}$, x is preferred over y in a majority of orderings.

The Smith set can be computed using standard graph based algorithms. By representing global preferences as directed edges in a graph (e.g., $x \to y$ if x is globally preferred over y), computing the Smith set is now equivalent to finding the strongly connected component with no incoming edges from outside of the component. The Smith set is therefore, sometimes referred to as the "top cycle." It can be found by using variations of the Floyd-Warshall algorithm or Tarjan's algorithm (see [13] for common graph algorithms). Similarly, all candidates can be ranked by topologically sorting the strongly connected components.

We note that it is not particularly difficult to compute the fair ordering in the static finalization problem. Indeed, the same techniques for computing the Smith set can be used to find the fair ordering for transactions. We detail the algorithm in Figure 1. When no Condorcet cycles exist in $\mathcal{L}$, this protocol solves the strong static finalization problem; otherwise, it solves the weak variant.

3.1.2 Streaming finalization.

The primary difference that separates the Aequitas finalization protocol from prior work in voting theory is that the list of preferences is not fully determined at the start of the protocol. In the consensus setting, the protocol is ongoing as new transactions are continuously input to nodes[7]. Clearly nodes should not have to wait for all transactions to be input to decide on the fair ordering. Here, as noted by KZGJ, future transactions can cause changes in the condensation graph. Similarly, current transactions may not have been seen sufficiently many times so far. This means that current transactions might need to wait for future ones in order to be output in a fair order. The

[7]This is analogous to a growing list of potential candidates in an election, which is not particularly relevant for voting theory.

key technical challenge is to identify when transactions can be delivered without compromising on their fair ordering. We formalize this as the streaming finalization problem (Problem 3.6).

For lists List and List', we use $\mathsf{List} \preccurlyeq \mathsf{List}'$ to denote that List is a prefix of List'. For $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_k]$ and $\mathcal{L}' = [\mathsf{List}'_1, \ldots, \mathsf{List}'_k]$, we also use the notation $\mathcal{L} \preccurlyeq \mathcal{L}'$ to denote that $\mathsf{List}_j \preccurlyeq \mathsf{List}'_j$ for all j.

Problem 3.6. (Streaming Fair-Order Finalization) Consider an admissible $\Upsilon = (n, f, \gamma)$. At each timestep r, consider as input $\mathcal{L}^r = [\mathsf{List}_1^r, \ldots, \mathsf{List}_n^r]$ such that $\mathcal{L}^{r_1} \preccurlyeq \mathcal{L}^{r_2}$ whenever $r_1 \leq r_2$. At timestep r, output a list $\mathsf{OutputList}^r$ such that:

- (Prefix-Consistency) $\mathsf{OutputList}^{r_1} \preccurlyeq \mathsf{OutputList}^{r_2}$ whenever $r_1 \leq r_2$.
- (Strong (resp. Weak) Fairness of ordering) For all (m, m') such that $m' \lhd^{\Lambda} m$, where $\Lambda = (\Upsilon, \mathcal{L}^r)$, if $\mathsf{OutputList}^r$ contains m or m', then m is ordered before (resp. no later than) m'.

We say that an algorithm $F(\cdot)$ solves the streaming finalization problem for an admissible Υ if for any valid input sequence $\mathcal{L}^1, \mathcal{L}^2, \ldots$, for every timestep r, $F(\mathcal{L}^r)$ satisfies prefix-consistency and fairness of ordering. Note that if $\mathcal{L} \preccurlyeq \mathcal{L}'$, then $m' \lhd^{(\Upsilon, \mathcal{L})} m$ implies $m' \lhd^{(\Upsilon, \mathcal{L}')} m$ since for any $\mathsf{List}_j \in \mathcal{L}$ where $m \prec m'$, the corresponding $\mathsf{List}'_j \in \mathcal{L}'$ will also have $m \prec m'$. In other words, the fairness of ordering condition will not contradict the prefix-consistency condition.

Remark. Note that liveness is not included in Problem 3.6 (analogous to how delivering no transactions still satisfies"consistency" in the consensus abstraction). Liveness will be a separate property for our protocols.

3.1.3 Connecting back to order-fairness

Given the n input orderings from the protocol nodes, the Aequitas finalization stage first builds a dependency graph of transactions where edges correspond to global preferences and computes its condensation graph. From the condensation graph, the transactions in a vertex v are delivered in one of two ways:

1. v is a source vertex and for all other vertices v', there is a path from v to v'
2. v is a source vertex and for any other vertex v' without a path from v, there is a common descendant of v and v' and either v has more descendants or v, v' have the same number of descendants but v is preferred in a previously agreed upon ordering relation (e.g., alphabetical ordering).

KZGJ showed that whenever m is globally preferred to m', the output of the Aequitas finalization will order m no later than m' in the general case, and m strictly before m' when in a setting where there are no Condorcet cycles. In turn, this implies that the finalization stage in the Aequitas protocol solves the streaming finalization problem where the input lists for honest nodes are the actual ordering of inputs from $\mathcal{Z}$. Note that we can abstract out the underlying protocol nodes and run the Aequitas finalization only on the lists. In general, given $\Upsilon = (n, f, \gamma)$ and $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_k]$, we will use $\mathsf{Graph}_{\Upsilon}(\mathcal{L})$ to denote the dependency graph and $\mathsf{Aequitas}_{\Upsilon}(\mathcal{L})$ to denote the resultant outputs of applying the finalization step of the Aequitas protocol on the lists in $\mathcal{L}$. We write $\mathsf{Graph}(\mathcal{L})$ and $\mathsf{Aequitas}(\mathcal{L})$ when Υ is clear from context.

Fact 3.7. *For an admissible* Υ*,* $\mathsf{Aequitas}_{\Upsilon}(\cdot)$ *solves the weak streaming fair-order finalization problem.*

Proof. First, from the self-consistency of the Aequitas protocol, we can infer that if $\mathcal{L} \preccurlyeq \mathcal{L}'$, then $\mathsf{Aequitas}_{\Upsilon}(\mathcal{L}) \preccurlyeq \mathsf{Aequitas}_{\Upsilon}(\mathcal{L}')$. Furthermore, $\mathsf{Aequitas}_{\Upsilon}(\cdot)$ satisfies the weak fairness of ordering property since the Aequitas protocol from KZGJ satisfies block-order-fairness in the permissioned setting. □

3.1.4 Player Replaceability

So far, the input lists we used for $\mathsf{Aequitas}(\mathcal{L})$ were the actual transaction input orderings (for honest nodes) in a permissioned protocol. The key idea we use to move to the permissionless setting is to enable nodes to "continue" the orderings of other nodes in the system. In particular, we will fix a number of input lists, say k, at the start of the protocol. All nodes now extend the existing transaction orderings in these k lists. This can be done through any permissionless consensus technique (e.g., proof-of-work, proof-of-stake etc). The k lists can be thought of as the input orderings of k "virtual" nodes in a permissioned network. The Aequitas finalization protocol will now be run on the k lists to determine the final output ordering. To allow for a seamless reduction of the permissionless problem to a permissioned one, we use the following player replaceability lemma. Intuitively, this lemma will directly let us conclude order-fairness for our permissionless protocols, by proving order-fairness for the permissioned transformation with k virtual parties. For our protocols, the Aequitas finalization will guarantee the order-fairness in the virtual party setting.

Lemma 3.8 (Player Replaceability)**.** *Consider an* $(\mathcal{A}, \mathcal{Z})$ *that satisfies* $(n, \beta, \Delta, \cdot, \cdot)$*-permissionless execution, an order-fairness parameter* γ*, an admissible* $\Upsilon = (k, k\beta', \gamma)$ *for some* $0 \leq \beta' \leq 1$*. Suppose that* Π *is a protocol as follows: For any node* N *that is honest in round* r*,* N *outputs* $\mathcal{L}_N^r = [\mathsf{List}_1^r, \ldots, \mathsf{List}_k^r]$ *to* $\mathcal{Z}$ *that satisfies the following:*

- *If* N *is also honest in round* $r-1$*, then* $\mathcal{L}_N^{r-1} \preccurlyeq \mathcal{L}_N^r$*.*
- *For any pair of transactions* $(\mathrm{tx}, \mathrm{tx}')$ *that satisfies the antecedent of the (permissionless)* γ*-order-fairness definition (Definition 2.3), if either* tx *or* tx' *is present in all* k *lists in* $\mathcal{L}_N^r$*, then* tx *will be* γ*-globally preferred to* tx' *w.r.t.* $\Lambda = (\Upsilon, \mathcal{L}_N^r)$*.*

  Note that we will have either transaction present in all (rather than $k - \beta k$*) lists, as honest nodes will also have the ability to append transactions to all* k *lists (e.g., through PoW mining).*

Suppose further that an algorithm $F(\cdot)$ *satisfies the streaming strong (resp. weak) fair-order finalization problem for* Υ *such that the* $\mathsf{linearize}$ *function of* Π *is defined to be* $F(\mathcal{L}_N^r)$*. Then,* Π *satisfies* γ*-receive-order-fairness (resp.* γ*-block-order-fairness) w.r.t.* $(\mathcal{A}, \mathcal{Z})$*.*

Proof. The proof is straightforward. For any pair $(\mathrm{tx}, \mathrm{tx}')$ that satisfies the antecedent of the (permissionless) γ-order-fairness definition, we are given that Π is such that tx will be γ-globally preferred to tx' in the k lists that represent the virtual parties. Now, since $F(\cdot)$ satisfies the strong (resp. weak) fairness of ordering property, the final output ledger of an honest node in the given round, will contain tx before (resp. no later than) tx'. □

Intuitively, given a protocol Π that satisfies the specified properties (i.e., it provides a transformation to k virtual parties), the upshot of this lemma is that now, we can run $F(\cdot) = \mathsf{Aequitas}(\cdot)$ on the

k lists to determine the final fair output ordering. The only remaining part now is to actually construct such a permissionless protocol Π. We will show two constructions in this paper: Π_{mod} (Section 5), where a single PoW chain will be modularized into k chains, each representing one virtual party, and Π_{multi} (Section 6) where k PoW chains will be mined simultaneously.

Remark on Liveness. The Aequitas protocol from KZGJ achieves (conventional) liveness when no cycles can exist in the dependency graph or if the cycles cannot extend for across some bounded time interval. One such setting is when the external network (in the permissioned setting) has $\Delta_{\mathsf{ext}} = 1$. When reducing the problem in the permissionless setting to running $\mathsf{Aequitas}(\cdot)$ on the k lists, we emphasize that this no longer corresponds to requiring $\Delta = 1$ in our permissionless setting. Instead, this is the assumption that the sequence number of any given transaction in all k lists differs by at most 1. In our protocols, this turns out to be false due to potential adversarial influence on all k input lists. Fortunately, when $\Delta = 1$ (in the permissionless environment), we can still show that any cycle that exists in the dependency graph of the k lists cannot extend for too long (Lemma 4.3). This will allow our protocols to achieve conventional liveness (see Section 4 for details).

4 Fair Ordering Protocols

We will provide two fair ordering protocols, Π_{mod} (Section 5) and Π_{multi} (Section 6) that each satisfy consistency, liveness, and block-order-fairness. Π_{mod} uses a single PoW Nakamoto chain. The Π_{multi} protocol uses multiple parallel chains instead with each chain serving to vote on transaction orderings.

4.1 Overview of Proof Techniques

Consistency. The consistency argument for our protocols will directly follow from the consistency of the underlying Nakamoto blockchain.

Order-Fairness. We start by defining a soundness property that will be sufficient to achieve order-fairness. Informally, there should be an edge from tx to tx′ in the dependency graph of any honest node, whenever (tx, tx') satisfies the antecedent of the order-fairness definition.

Property 4.1 (Soundness of Dependencies)**.** Consider $(n, \beta, \Delta, \tau, R)$ and an order-fairness parameter γ. We say that Π is γ-*dependency-sound* if for any $(\mathcal{A}, \mathcal{Z})$ that satisfies $(n, \beta, \Delta, \tau, R)$-permissionless execution, the following property holds: If (tx, tx') satisfies the antecedent of the γ-order-fairness definition, then for any round r, and any honest node N in round r, if N outputs $\mathcal{L}$ where tx′ is in all lists in $\mathcal{L}$, then $\text{tx}' \lhd^{\Lambda} \text{tx}$ holds where $\Lambda = (\Upsilon = (|\mathcal{L}|, \lceil \frac{|\mathcal{L}|}{2} \rceil - 1, 1), \mathcal{L})$.

We show that our protocols satisfy Property 4.1. It is not difficult to see how this will imply order-fairness through the player replaceability lemma (Lemma 3.8). Consider a protocol Π that is γ-dependency sound, and a transaction pair (tx, tx') that satisfies the antecedent of the γ-order-fairness definition. This means that if an honest node outputs $\mathcal{L}$ where tx′ is contains in all lists, the dependency graph of $\mathcal{L}$ will contain the edge $\text{tx} \to \text{tx}'$. Now, since the function $\mathsf{linearize}(\cdot)$ (which decides the final agreed upon ledger) is defined to be the output of $\mathsf{Aequitas}$ on $\mathcal{L}$, and $\mathsf{Aequitas}(\cdot)$

satisfies the streaming finalization problem (i.e., it will output tx no later than tx′), this allows us to directly use the player-replaceability lemma (Lemma 3.8) to conclude block-order-fairness.

Remark (Receive-Order-Fairness). KZGJ provided a synchronous permissioned protocol that achieves receive-order-fairness when the synchrony parameter in the external network (between clients and nodes) was $\Delta_{\mathsf{ext}} = 1$. This is in part because an adversary can no longer claim to have received transactions too far apart from honest users (otherwise they would be easily detected), thereby resulting in no Condorcet cycles. For our purpose, since the adversary's orderings become intermingled with the honest ones through the PoW mining, attempting to realize receive-order-fairness would require rewinding PoW chains which might lead to other complications.

Although the impossibility result for receive-order-fairness from KZGJ when $\Delta_{\mathsf{ext}} > 1$ (and the adversary is allowed to corrupt at least one party) carries over to the permissionless setting, we leave it an upon problem to resolve whether receive-order-fairness is achievable in the permissionless setting when $\Delta = 1$.

Remark (Zero-Block-Confirmation when $\Delta = 1$). Although our protocols achieve block-order-fairness (and not receive-order-fairness), when $\Delta = 1$, as we show in Lemma 4.3, for a transaction tx that is first received in round r, any Condorcet cycle will extend only till $r+1$. This means that any transaction that first arrives after this will be ordered strictly after tx in the final ledger. This will also allow us to achieve liveness, as well as zero-block-confirmation (see Section 7).

Liveness. To show liveness, we will show the following faithfulness property for our protocols. Informally, a protocol is *dependency-faithful* if some honest node has received tx before tx′ whenever an edge from tx to tx′ exists in the dependency graph of an honest node. Now, if a protocol is dependency-faithful, then when $\Delta = 1$, we show that if transactions tx and tx′ are in the same Condorcet cycle, then they cannot have been received far apart. In other words, a transaction does not have to wait for an arbitrary length of time to ensure that it does not get combined with a later transaction in the same strongly connected component of the dependency graph. Consequently, this will enable transactions to be delivered in a bounded length of time satisfying liveness. Note that the specific liveness bound will be dependent on the actual protocol.

Property 4.2 (Faithfulness of Dependencies)**.** Consider $(n, \beta, \Delta, \tau, R)$. We say that Π is *dependency-faithful* if for any $(\mathcal{A}, \mathcal{Z})$ that satisfies $(n, \beta, \Delta, \tau, R)$-permissionless execution, the following property holds: For any round r, and any honest node N in round r, if N outputs $\mathcal{L}$ with $\mathrm{tx}' \lhd^{\Lambda} \mathrm{tx}$ where $\Lambda = (\Upsilon = (|\mathcal{L}|, \lceil \frac{|\mathcal{L}|}{2} \rceil - 1, 1), \mathcal{L})$, then there is at least one honest node that has received tx before tx′ as input from $\mathcal{Z}$. In other words, if an edge $\mathrm{tx} \to \mathrm{tx}'$ exists in $\mathsf{Graph}_{\Lambda}(\mathcal{L})$, then there is some honest node that received tx before tx′ as input.

We show in Lemma 4.3 that when $\Delta = 1$, for any protocol that is dependency-faithful, transactions (even adversarial ones) cannot be in the same Condorcet cycle if they are received far apart. This will prevent transactions for "waiting" too long in the dependency graph, and allow us to conclude liveness.

Lemma 4.3. *Consider* $(n, \beta, \Delta = 1, \tau, R)$ *and suppose that a protocol* Π *is dependency-faithful (Property 4.2). Suppose that some honest node in round* r *outputs* $\mathcal{L}$ *where* tx *and* tx′ *are in the same Condorcet cycle. Let* r, r' *be the first round that some honest node has received* tx *and* tx′ *respectively. Then* $r = r'$.

Proof. Let $(\mathrm{tx}_1, \mathrm{tx}_2, \ldots, \mathrm{tx}_l)$ be a Condorcet cycle in $\mathcal{L}$ and suppose that tx and tx$'$ are in this cycle. Let round r_{first} be the first round that any one of $\{\mathrm{tx}_1, \ldots, \mathrm{tx}_l\}$ was received by an honest node. We can assume tx_l was first received by an honest node at round r_{first} without loss of generality by rotating the tuple cyclically. This means that any honest node that spawned no later than $r_{\text{first}} + 1$, received tx_l either in round r_{first} or $r_{\text{first}} + 1$, and any honest node that spawned in round $r' > r_{\text{first}} + 1$ received tx_l in round r'.

Now, suppose that tx_{l-1} was first received by an honest node at round r_{l-1} and suppose for the sake of contradiction that $r_{l-1} \geq r_{\text{first}} + 1$. This means that any honest node that spawned no later than r_{l-1} had already received tx_l (i.e., either tx_l before tx_{l-1} or tx_l and tx_{l-1} in the same round) and any node that spawned after received both tx and tx$'$ at the same time. However, since we have $\mathrm{tx}_l \lhd \mathrm{tx}_{l-1}$, this contradicts the dependency-faithfulness of Π since there needs to be some honest node that received tx before tx$'$. Furthermore, since no transaction in $\{\mathrm{tx}_1, \ldots, \mathrm{tx}_l\}$ was received before r_{first}, this means that tx_{l-1} was first received by an honest node in round r_{first}.

Continuing in the same fashion, the same argument shows that any transaction in $\{\mathrm{tx}_1, \ldots, \mathrm{tx}_l\}$, including tx and tx$'$, was first received by an honest node in round r_{first}.

Note that also applies to the case where some of the transactions are adversarial (i.e. first seen by the network as part of a block rather than transaction gossip). Recall that the ledger synchrony assumption of our network ensures that transactions are provided as input to all honest nodes if some honest node outputs them to the environment. □

Corollary 4.3.1 (Informal)**.** *Consider a protocol* Π *that is dependency-faithful (Property 4.2) and suppose that* $\mathsf{Aequitas}(\cdot)$ *is used as the* $\mathsf{linearize}(\cdot)$ *function. If* $\Delta = 1$*, then* Π *will satisfy liveness (for some parameter).*

Proof (Sketch). This is a direct consequence of the previous lemma. Since Condorcet cycles do not extend for long (i.e., transactions cannot be blocked waiting for other transactions arriving much later), $\mathsf{Aequitas}(\cdot)$ will be able to deliver transactions from the dependency graph "quickly". The exact liveness parameter will of course, depend on the specific protocol. □

5 Modulo-T Longest Chain Protocol

We start with a basic order-fair protocol $\Pi_{\text{mod}}^{\kappa,T}$ that uses a single PoW-style longest chain, where κ is the security parameter, and T is the modular parameter. While Π_{mod} mines only a single PoW chain, blocks in the same index modulo T will be considered to be semantically or logically connected. For example, blocks at indices 1, $T + 1$, $2T + 1$, and so on will be part of the same "modular chain."

Protocol description. In $\Pi_{\text{mod}}^{\kappa,T}$, honest nodes mine a single PoW chain. For a given chain, the block $\mathsf{chain}[d]$ is considered to logically follow the block $\mathsf{chain}[d - T]$, despite its PoW hash following the block $\mathsf{chain}[d - 1]$. This divides the single PoW chain into T modular chains. Any transaction will be included once in each modular chain. Abstractly, the T modular chains will define T separate transaction orderings which will be used as input for the $\mathsf{Aequitas}(\cdot)$ algorithm to retrieve the final transaction ordering.

For a given chain, let $\mathcal{T}_d$ be the set of transactions that were first seen in block $\mathsf{chain}[d]$. Now, if $\mathsf{chain}[d]$ was honestly mined, then the first honestly mined block after $\mathsf{chain}[d]$ in each modular

$\Pi^{\kappa,T}_{\text{mod}}(p)$

Initialize: $\mathsf{chain} = \mathit{genesis}$; $\mathsf{TxPool} = \emptyset$

Upon receiving a valid chain' such that $|\mathsf{chain}'| > |\mathsf{chain}|$:

- Set $\mathsf{chain} := \mathsf{chain}'$ and output any newly seen transactions to $\mathcal{Z}$.

In round r, upon receiving txs from $\mathcal{Z}$:

- For each tx in txs, add (tx, r) to TxPool.
- Set h_{-1} to be the reference of $\mathsf{chain}[-1]$ and h' to be the reference of $\mathsf{chain}[-T]$ if $|\mathsf{chain}| > T$ otherwise set h' to be the reference of $\mathsf{chain}[0]$. Suppose that $\mathsf{chain}[-1]$ belongs to modular chain d.
- Let $\mathsf{txincludes}$ be the transactions in TxPool that are not any block in modular chain $(d+1 \bmod T)$, ordered by increasing round number.
- For $i = 1$ to q:
  - Pick a random nonce η and let $h = \mathsf{H}(h_{-1}, h', \mathsf{txincludes}, \eta)$.
  - If $h < D_p$, then set $\mathsf{chain} := \mathsf{chain} \parallel (h_{-1}, h', \mathsf{txincludes}, \eta, h)$, broadcast chain and break.
- Let $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_T] = \mathsf{modularize}(\mathsf{chain}[:-T], T)$
- Output $\mathcal{L}$ to $\mathcal{Z}$. The final linearly ordered log output is taken as $\mathsf{linearize}(\mathcal{L})$.

$\mathsf{linearize}(\mathcal{L})$

- Output $\mathsf{Aequitas}(\mathcal{L})$.

Figure 2: Protocol $\Pi^{\kappa,T}_{\text{mod}}$

chain should contain these transactions. In other words, all honest nodes can reject blocks in the range $\mathsf{chain}[d+1, d+T-1]$ that do not contain all transactions in $\mathcal{T}_d$. Consequently, in a valid chain, the record $\mathtt{m}$ in any block $\mathsf{chain}[d]$ should be such that $\bigcup_{i=d-T+1}^{d-1} \mathcal{T}_i \subseteq \mathtt{m}$.

Figure 2 contains a detailed description of Π_{mod}.

Valid chain. A chain is valid iff (1) $\mathsf{chain}[0] = \mathit{genesis} = (0, 0, \perp, 0, H(0, 0, \perp, 0))$ is the genesis block; (2) $\mathsf{chain}[d].h_{-1} = \mathsf{chain}[d-1].h$ for all $d \in \{1, \cdots, |\mathsf{chain}|\}$; (3) $\mathsf{chain}[d].h' = \mathit{genesis}.h$ for $d \in \{1, \ldots, T\}$ and $\mathsf{chain}[d-T].h$ for all $d \in \{T+1, \cdots, |\mathsf{chain}|\}$; and (4) $\bigcup_{i=d-T+1}^{d-1} \mathcal{T}_i \subseteq \mathsf{chain}[d].\mathtt{m}$ for $d \in \{1, \cdots, |\mathsf{chain}|\}$.

Modular chains. For a given valid chain, we say that two blocks $\mathsf{chain}[b]$ and $\mathsf{chain}[b']$ where b, b' are positive integers are in the same modulo-T chain if $b \equiv b' \mod T$. The genesis block, $\mathsf{chain}[0]$ is assumed to be included in all modular chains. We use $\mathsf{modularize}(\mathsf{chain}, T)$ to denote the operation of splitting chain into modulo-T chains. Specifically, the output of $\mathsf{modularize}(\mathsf{chain}, T)$ is $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_T]$ such that List_j includes transactions in all blocks $\mathsf{chain}[b]$ where $b \equiv j \mod T$ in a (partial) ordering determined by increasing block number. We will use $\mathtt{ref} = h'$ to point to the previous block in the same modular chain.

Dynamic corruption. As mentioned earlier, our protocols do not support a fully adaptive adversary. Instead, we restrict our adversary to be (τ, R)-respawning where at most τn nodes can

be killed within any R round period. For the protocol $\Pi_{\text{mod}}^{\kappa,T}$, we will choose R such that, except with negligible probability, R is larger than the time it takes for a transaction tx to be confirmed in all modular chains. It suffices to chose R larger than the time to mine $\frac{5T}{2}$ blocks (T blocks that contain tx, T blocks for confirmation plus $T/2$ possible adversarial blocks before the first time tx is mined).

Consider an $(\mathcal{A}, \mathcal{Z})$ that is $(n, \beta, \Delta, \tau, R)$-permissionless where $\beta + \tau < \frac{1}{3}$. For a given pair (tx, tx') of transactions, suppose that round r is the first round that the n nodes online in round r have received either tx or tx$'$ and let $\mathcal{N}_{(\text{tx},\text{tx}')}$ be the set of those nodes. Note that at most βn of those are corrupt and the rest (at least $n - \beta n$) are honest. Now, suppose that γ fraction of nodes have received tx before tx$'$. This means that at least $\gamma n - \beta n$ honest nodes (at round r) received tx before tx$'$. Of these, at least $\gamma n - \beta n - \tau n$ will be honest at the time at least one of tx and tx$'$ has been mined into all modular chains, since we restrict the respawning of our adversaries. We will show that if $\gamma - (\beta + \tau) > \frac{2}{3}$, then more than half of the blocks that mine tx or tx$'$ first in a modular chain (i.e. the blocks that decide on the ordering between tx and tx$'$) will be mined by an honest node that received tx before tx$'$ from the environment. This will directly imply that $\Pi_{\text{mod}}^{\kappa,T}$ is dependency-sound allowing us to conclude order-fairness.

Compliant parameters. We define a compliance predicate for $\Pi_{\text{mod}}(p)$ as $\Gamma_{\text{mod}}^{p}(n, \beta, \Delta, \gamma, T, \tau, R) = 1$ if n, Δ, T, R are polynomials in κ; the parameters β, γ, τ are constants such that $\gamma - (\beta + \tau) > \frac{2}{3}$; and for all κ, $\Gamma_{\text{nak}}^{p}(n, \beta, \Delta) = 1$, $\Pi_{\text{nak}}(p)$ satisfies T-common-prefix and $(T, \mu > \frac{1}{2})$-chain-quality, $R \geq \frac{5T+2}{2g_0}$ where g_0 is such that $\Pi_{\text{nak}}(p)$ also satisfies $(T, g_0, \cdot)$-chain-growth.

5.1 Security Proofs

We first note that consistency of $\Pi_{\text{mod}}^{\kappa,T}$ comes for free from the consistency property of Nakamoto's blockchain since we use a single PoW chain. To prove order-fairness, we will prove that Π_{mod} is dependency-sound (Property 4.1). In other words, tx is globally preferred to tx$'$ in the lists $\mathcal{L}$ whenever (tx, tx') satisfies the antecedent of the order-fairness definition. This allows us to conclude order-fairness using the player replacability lemma (Lemma 3.8). For liveness (when $\Delta = 1$), we will show that Π_{mod} is dependency-faithful (Property 4.2).

We start by showing that Π_{mod} is dependency-sound in the following lemma.

Lemma 5.1. *Consider parameters that satisfy $\Gamma_{\text{mod}}^{p}(n, \beta, \Delta, \gamma, T, \tau, R) = 1$. Then, $\Pi_{\text{mod}}^{\kappa,T}(p)$ is γ-dependency-sound for $(n, \beta, \Delta, \tau, R)$ environments.*

Proof. Consider any $(\mathcal{A}, \mathcal{Z})$ that satisfies $(n, \beta, \Delta, \tau, R)$-permissionless execution. Consider a pair of transactions (tx, tx') that satisfies the antecedent of the order-fairness definition. For a given chain, suppose that $\mathsf{chain}[d]$ is the first block that contains either tx or tx$'$ and that $|\mathsf{chain}| \geq d + 2T - 1$. Note that all blocks in $\mathsf{chain}[d, d + T - 1]$, and consequently all modular chains, will include either tx or tx$'$. Now, since $\gamma - (\beta + \tau) > \frac{2}{3}$ and $\mathcal{A}$ is (τ, R)-respawning, there are more than $\frac{2n}{3}$ nodes that received tx before tx$'$ and are honest until all modular chains contain either tx or tx$'$ (i.e. until $\mathsf{chain}[d+T-1]$ has been confirmed). Let H be the set of these nodes. Now, from the chain-quality of $\Pi_{\text{nak}}(p)$, in the range $\mathsf{chain}[d, d+T-1]$, more than $\frac{T}{2}$ of these blocks are mined by the nodes in H. In other words, more than half of the blocks in $\mathsf{chain}[d, d+T-1]$ will contain tx before tx$'$. Since these blocks have been confirmed in chain, more than half of the lists in $\mathcal{L} = \mathsf{modularize}(\mathsf{chain}[: -T], T)$

will contain tx before tx′. Since $\Upsilon = (T, \lceil \frac{T}{2} \rceil - 1, 1)$ is admissible, tx is globally preferred to tx′ w.r.t. $(\Upsilon, \mathcal{L})$, i.e. $\text{tx}' \lhd^{(\Upsilon,\mathcal{L})} \text{tx}$.
We conclude that $\Pi_{\text{mod}}^{\kappa,T}$ is dependency-sound. □

Theorem 5.2 (Order-Fairness of Π_{mod})**.** *Consider parameters that satisfy* $\Gamma_{\text{mod}}^{p}(n, \beta, \Delta, \gamma, T, \tau, R) = 1$. *Then,* $\Pi_{\text{mod}}^{\kappa,T}(p)$ *satisfies* γ*-block-order-fairness.*

Proof. This is a straightforward combination of Fact 3.7, Lemma 3.8, and Lemma 5.1. Note that for the simplest case of $\gamma = 1$, we require $\beta + \tau < 1/3$. □

Next, we show that Π_{mod} is dependency-faithful in the following lemma.

Lemma 5.3. *Consider parameters that satisfy* $\Gamma_{\text{mod}}^{p}(n, \beta, \Delta, \gamma, T, \tau, R) = 1$. *Then,* $\Pi_{\text{mod}}^{\kappa,T}(p)$ *is dependency-faithful for* $(n, \beta, \Delta, \tau, R)$ *environments.*

Proof. Consider any $(\mathcal{A}, \mathcal{Z})$ that satisfies $(n, \beta, \Delta, \tau, R)$-permissionless execution. Suppose that a node N is honest in round r, holds chain, and outputs $\mathcal{L} = \mathsf{modularize}(\mathsf{chain}[:-T], T)$ to $\mathcal{Z}$. Note that $\mathcal{L}$ only contains the confirmed part of each modular chain. Let $\Upsilon = (T, \lceil \frac{T}{2} \rceil - 1, 1)$ and $\Lambda = (\Upsilon, \mathcal{L})$. Suppose that $\text{tx}' \lhd^{\Lambda} \text{tx}$, and let $\mathsf{chain}[d]$ be the first block that contains either tx or tx′. Then, the range $\mathsf{chain}[d, d + T - 1]$ will determine the ordering between tx and tx′ in each of the T modular chains. Since $\text{tx}' \lhd^{\Lambda} \text{tx}$, more than $\frac{T}{2}$ of these blocks order tx before tx′. But, since $\beta < \frac{1}{3}$ (from $\gamma - (\beta + \tau) > \frac{2}{3}$), the chain-quality property of $\Pi_{\text{nak}}(p)$ implies that less than half of the blocks in the range $\mathsf{chain}[d, d + T - 1]$ are mined by an adversary. This means that there is at least 1 block that contained tx before tx′ which was mined by an honest node. Therefore, some honest node received tx before tx′ from $\mathcal{Z}$.
We conclude that $\Pi_{\text{mod}}^{\kappa,T}$ is dependency-faithful. □

Theorem 5.4 (Liveness of Π_{mod})**.** *Consider parameters that satisfy* $\Gamma_{\text{mod}}^{p}(n, \beta, \Delta = 1, \gamma, T, \tau, R) = 1$*, and suppose that* $\Pi_{\text{nak}}(p)$ *satisfies* (T, g_0, g_1)*-chain-growth. Then* $\Pi_{\text{mod}}^{\kappa,T}(p)$ *satisfies* $(5T+2)/(2g_0)$*-liveness.*

Proof. Consider any transaction tx that is first input in round r. By the time more than $\frac{5T}{2}$ blocks are mined, all semantic chains will contain tx in the confirmed part of the chain as well as any transactions that are in the same cycle as tx. Furthermore, any transaction that is first received by an honest node at $r+2$ will be a descendant of tx (as well as any other tranasction without an edge to/from tx) in the dependency graph of an honest node. Note that such a transaction will also be in the confirmed part of the chain. Consequently, by this time $\mathsf{Aequitas}(\cdot)$ will have delivered tx.

Now, we note that for a round r' such that $r' - r \geq \frac{5T+2}{2g_0}$, we have $g_0(r' - r) \geq \frac{5T}{2} + 1$, which implies that tx will be output by round r' by all honest nodes. Consequently, Π_{mod} satisfies $\frac{5T+2}{2g_0}$-liveness. □

Remark. We can handle minority corruption by using Fruitchains [34] as the underlying consensus protocol. Fruitchains achieves optimal mining-fairness, which means that for any $\beta < \frac{1}{2}$, and a sufficiently long period T, it has $(T, \mu_0 > \frac{1}{2})$-chain-quality. Consequently, by using Fruitchains, our construction can tolerate $\beta + \tau < \frac{1}{2}$ corruption and still achieve order-fairness (for $\gamma = 1$).

Our construction can also be layered on top of other longest-chain protocols that satisfy the chain-quality property (e.g., Proof-of-Stake protocols like [27]).

6 Multi-Chain Protocol

In this section, we introduce the protocol $\Pi^{\kappa,g}_{\text{multi}}$ (where g is the parallelization parameter), that uses multiple parallel Nakamoto chains. Intuitively, in $\Pi^{\kappa,g}_{\text{multi}}$, honest nodes will mine g PoW chains in parallel, with each of the parallel chains serving as an ordering for transactions. Similar to the single-chain protocol, $\mathsf{Aequitas}(\cdot)$ will now be run on the confirmed parts of each of the g chains i.e., each chain with the last T blocks cut off, where $T = \omega(\log \kappa)$ is the consistency parameter of the underlying Nakamoto chains. This will allow us to retrieve the fair ordering from the g chains. Figure 3 contains a complete description of our protocol. We will restrict the order-fairness parameter to only $\gamma = 1$, and our network delay parameter to $\Delta = 1$. Following [35], we will also use $p = \Theta(1/n)$ to make the per round mining rate $f = pqn$, be $\Theta(1)$ in κ.

g-for-1 PoW mining. To mine blocks, we use a g-for-1 mining technique, similar to the one from [7], which generalizes the 2-for-1 trick from [22, 34]. Abstractly, nodes will simultaneously mine transactions on all g chains, and the chain to which the block gets appended to will depend on the hash value.

For this, we will slightly abuse the notation for blocks from Section 4. Specifically, a block will be a tuple $(\mathbf{h}_{-1}, \mathbf{h}', \mathbf{m}, \eta, h)$ where $\mathbf{h}_{-1}$ will be a list of parent blocks on all g chains, $\mathbf{h}'$ will be a list of reference blocks on all g chains, and $\mathbf{m}$ is the list of transactions to be (potentially) appended to each chain. This is called a *superblock* in [7] but for simplicity, we use the same formalism of a block. For our purpose, we will have $\mathbf{h}_{-1} = \mathbf{h}'$. We will use a random oracle H that outputs $g\kappa$ bits (instead of κ). For hardness parameter $p(\kappa)$, define $D_p = p(\kappa) \cdot 2^{\kappa}$. For a block with hash $h = \mathsf{H}(\mathbf{h}_{-1}, \mathbf{h}', \mathbf{m}, \eta)$, we will say that it is "mined into chain i", if $h(i) < D_p$ where $h(i)$ is the substring of h in range $(i-1)\kappa$ to $i\kappa - 1$. Considering different output bits for different chains allows us to make the mining for each chain independent although it does increase the size of the random oracle output. A block can be propagated to the rest of the network as part of the chain it gets mined in. For simplicity, we assume that the entire block is propagated, but note that in practice, similar to [7, 34], only the data relevant to its chain can be included, along with a commitment to the other discarded data.

Admissible parameters. We define a compliance predicate $\Gamma^{p}_{\text{multi}}(n, \beta, \Delta = 1, \gamma = 1, \tau, R, g, W) = 1$ if n, R, W are polynomials in κ with $p = \Theta(1/n)$; β, τ are constants such that $\beta + \tau < \frac{7-\sqrt{17}}{16}$; and for all κ, $\Gamma^{p}_{\text{nak}}(n, \beta, \Delta) = 1$, and there exists $T = T(\kappa)$ such that $\Pi_{\text{nak}}(p)$ satisfies T-common-prefix and $(T, \mu > \frac{1}{2})$-chain-quality, and $R(\kappa) \geq \frac{3T+2}{2g_0}$ where g_0 is such that $\Pi_{\text{nak}}(p)$ also satisfies $(T, g_0, \cdot)$-chain-growth. Furthermore, $W = \Theta(\kappa)$ and $g = \Theta(\kappa)$.

We will require R to be longer than the time it takes to mine $\frac{3T}{2}$ blocks in each chain. This will allow for tx to be mined and confirmed in all chains. W denotes the warmup time of our protocol.

6.1 Overview of security proofs

We will prove security for settings when $\Delta = 1$, and use $\gamma = 1$ for our order-fairness parameter. We define cross-chain-quality as Property 6.1 and provide a sketch of our proof that Π_{multi} satisfies it in this section. We defer the full proof to Appendix A.

$\Pi^{\kappa,g}_{\text{multi}}(p)$

Initialize: $\mathcal{C} = [\mathsf{chain}_1, \ldots, \mathsf{chain}_g] = [genesis_1, \ldots, genesis_g]$; $\mathsf{TxPool} = \emptyset$; $T = T(\kappa)$ which is $\omega(\log \kappa)$

Upon receiving a valid $\mathcal{C}' = [\mathsf{chain}_1, \ldots, \mathsf{chain}_g]$:

- For all indices j, if $\left|\mathsf{chain}'_j\right| > |\mathsf{chain}_j|$, then set $\mathsf{chain}_j := \mathsf{chain}'_j$ and output any newly seen transactions to $\mathcal{Z}$.

In round r, upon receiving txs from $\mathcal{Z}$:

- For each tx in txs, add (tx, r) to TxPool.
- For all indices $j \in \{1, \ldots, g\}$:
  - Set both h^j_{-1} and h'_j to be the reference of $\mathsf{chain}_j[-1]$.
  - Let $\mathsf{txincludes}_j$ be the transactions in TxPool that are not any block in chain chain_j, ordered by increasing round number.
- Define $\mathbf{h}_{-1} = [h^1_{-1}, \ldots, h^g_{-1}]$, $\mathbf{h}' = [h'_1, \ldots, h'_g]$, and $\mathbf{txincludes} = [\mathsf{txincludes}_1, \ldots, \mathsf{txincludes}_g]$.
- For $i = 1$ to q:
  - Pick a random nonce η and let $h = \mathsf{H}(\mathbf{h}_{-1}, \mathbf{h}', \mathbf{txincludes}, \eta)$.
  - If for some j, $h(j) < D_p$, then set $\mathsf{chain}_j := \mathsf{chain}_j \parallel (\mathbf{h}_{-1}, \mathbf{h}', \mathbf{txincludes}, \eta)$
  - If a nonce solution was found, then broadcast $\mathcal{C}$ and break.
- Let $\mathcal{L} = [\mathsf{List}_1, \ldots, \mathsf{List}_g] = \mathsf{extract}(\mathcal{C})$
- Output $\mathcal{L}$ to $\mathcal{Z}$. The final linearly ordered log output is taken as $\mathsf{linearize}(\mathcal{L})$.

$\mathsf{extract}(\mathcal{C} = [\mathsf{chain}_1, \ldots, \mathsf{chain}_g])$

- Initialize all $\mathcal{L}_j$ to be empty.
- For all index $j \in \{1, \ldots, g\}$, and then for all $l < |\mathsf{chain}_j| - T$
  - Suppose that $\mathsf{chain}_j[l] = (\mathbf{h}_{-1}, \mathbf{h}', \mathbf{txincludes}, \eta, h)$. Append $\mathsf{txincludes}_j$ to $\mathcal{L}_j$.

$\mathsf{linearize}(\mathcal{L})$

- Output $\mathsf{Aequitas}(\mathcal{L})$.

Figure 3: Protocol $\Pi^{\kappa,g}_{\text{multi}}$

Cross-chain-quality. Lemma 6.2 shows that for Π_{multi} to be both dependency-sound and dependency-faithful, it is sufficient for it to satisfy the following *cross-chain-quality* property. Consequently, by showing cross-chain-quality, we directly get both 1-block-order-fairness and liveness for Π_{multi}. For our choice of W and R, using the same argument as Theorem 5.4 gives us $(W(\kappa), R(\kappa))$-liveness.

Property 6.1 (Cross-Chain-Quality)**.** For a given view and tx, let $\mathcal{B}_{\text{tx}}$ be the set of blocks that contain tx for the first time in the confirmed part (i.e. $\mathsf{chain}_j[: -T]$) of each of the g chains. Then, more than half of the blocks in $\mathcal{B}_{\text{tx}}$ were mined by honest nodes.

Lemma 6.2. *Consider $(n, \beta, \Delta = 1, \tau, R)$, and g, and define $\Pi^{\kappa,g}_{\text{multi}}(p)$ as before. Suppose that for all $(\mathcal{A}, \mathcal{Z})$ that support $(n, \beta, \Delta = 1, \tau, R)$-permissionless-execution, $\Pi^{\kappa,g}_{\text{multi}}(p)$ satisfies cross-*

chain-quality (except for a negligible number of views), then it is also 1-dependency-sound and dependency-faithful.

Proof.

1. (Dependency-Soundness) Suppose that $\Pi^{\kappa,g}_{\text{multi}}(p)$ satisfies cross-chain quality, and suppose for contradiction, that it is not dependency sound. Therefore, there is a pair of transactions $(\text{tx}_1, \text{tx}_2)$ that satisfies the antecedent of γ-order-fairness but an honest node outputs $\mathcal{L}$ with $\text{tx}_2 \lhd^{\Lambda} \text{tx}_1$ where $\Lambda = (\Upsilon, \mathcal{L})$ and $\Upsilon = (g, \lceil \frac{g}{2} \rceil - 1, 1)$.

   Now, for each chain, we consider 2 possibilities for the placement of the two transactions: either tx_1 is included before tx_2 or tx_2 is included before tx_1 (The case of both included at the same place is not relevant for our purpose). Furthermore, there are 4 possibilities for mining of the blocks containing tx_1, tx_2: both honest, both adversarial, only tx_1 honest, only tx_2 honest. We use the following notation to denote these 8 combinations: (H_1, A_2) represents that tx_1 was included before tx_2 and the block containing tx_1 was mined by an honest node while the block containing tx_2 was mined by an adversarial node. Let $\mathsf{num}(H_1, A_2)$ be the number of chains that have this combination.

   Since, $\text{tx}_2 \lhd^{\Lambda} \text{tx}_1$, more than $\frac{g}{2}$ should have included tx_2 before tx_1. Therefore,

$$\mathsf{num}(H_2, H_1) + \mathsf{num}(H_2, A_1) + \mathsf{num}(A_2, H_1) + \mathsf{num}(A_2, A_1) > \frac{g}{2}$$

   But since all honest nodes received tx_1 before tx_2 during the initial propagation, and $\Delta = 1$, any honest node that spawns later will receive tx_1 as soon as it spawns and therefore cannot have received tx_2 before tx_1. This means that, $\mathsf{num}(H_2, H_1)$ and $\mathsf{num}(H_2, A_1)$ will both be zero since no honest node will include tx_2 before tx_1. Therefore,

$$\mathsf{num}(A_2, H_1) + \mathsf{num}(A_2, A_1) > \frac{g}{2}$$

   But this contradicts cross-chain-quality since the adversary was able to mined tx_2 in more than half of the chains. This completes the proof.

2. (Dependency-Faithfulness) Suppose that $\Pi^{\kappa,g}_{\text{multi}}(p)$ satisfies cross-chain quality, and suppose for contradiction, that it is not dependency-faithful. Therefore, there is an honest node that outputs $\mathcal{L}$ with $\text{tx}_2 \lhd^{\Lambda} \text{tx}_1$ where $\Lambda = ((g, \lceil \frac{g}{2} \rceil - 1, 1), \mathcal{L})$ but there is no honest node that received tx_1 before tx_2.

   Now, using the same notation from the dependency-soundness argument, we have

$$\mathsf{num}(H_1, H_2) + \mathsf{num}(H_1, A_2) + \mathsf{num}(A_1, H_2) + \mathsf{num}(A_1, A_2) > \frac{g}{2}$$

   Since there is no honest node that received tx_1 before tx_2 (including nodes that spawn later since $\Delta = 1$), both $\mathsf{num}(H_1, H_2) + \mathsf{num}(H_1, A_2)$. Therefore, $\mathsf{num}(A_1, H_2) + \mathsf{num}(A_1, A_2) > \frac{g}{2}$, which once again will contradict cross-chain-quality for tx_1.

□

Proving cross-chain-quality. Now, to show that Π_{multi} satisfies the cross-chain-quality property, informally we proceed as follows. First, for a chain i and round r, we construct an event $V_i[r]$ such that whenever $V_i[r]$ holds, any transaction that is first seen by honest nodes at round r will be mined into an honest block. Next, we show that for a sufficiently small hardness parameter, the probability that this event happens, i.e., the transaction is mined into an honest block is more than 0.5. This means that by setting the number of chains to be large enough, we can now use the Chernoff bound to conclude that the transaction is contained in honestly mined blocks in more than half the chains, except with negligible probability. Equivalently, the cross-chain-quality property will be satisfied, except with negligible probability. Our proof makes novel use of techniques from the queuing theory literature and may be of independent interest. We point the reader to [19, 23] for a useful primer on the techniques we use.

To bound the probability for the event $V_i[r]$, we break it into two independent events $V_i^1[r]$ and $V_i^2[r]$ as follows:

- $V_i^1[r]$ is the event that in round r, there is no private adversarial chain that is longer than the chain i.
- $V_i^2[r]$ is the event that the first block mined in chain i after round r is honest and the block persists in the chain forever (except with negligible probability).

At a high level, under the event $V_i[r]$, if a transaction is first seen in round r, then it will be contained in the first honest block mined after round r in chain i and it will persist. Now, we use techniques from queuing theory to compute the probabilities for each event. Specifically, we start by formulating the event $V_i^1[r]$ as a discrete-time Markov chain (DTMC) and show it to be positive recurrent and irreducible. This allows us to use the ergodic theorem [23] and z-transforms to compute the probability that the event holds. For $V_i^2[r]$, we first model the event as a one-dimensional random-walk and then use z-transforms to compute its probability.

We defer the full proof to Appendix A.

Equipped with this, we can directly conclude the following theorem.

Theorem 6.3. *Consider parameters satisfying* $\Gamma^p_{\text{multi}}(n, \beta, \Delta = 1, \gamma, \tau, R, g, W) = 1$. *Then,* $\Pi^{\kappa,g}_{\text{multi}}(p)$ *satisfies* 1*-block-order-fairness, and* (W, R)*-liveness for* $(n, \beta, n, \Delta = 1, \tau, R)$ *environments.*

Liveness. Suppose that tx is first received by an honest node in round r. Since the chains are mined in parallel, we can simply consider the time it takes for any one chain. If $\Pi_{\text{nak}}(p)$ satisfies (T, g_0, g_1)-chain-growth, then for round r' such that $r' - r \geq \frac{3T+2}{2g_0}$, we have $g_0(r' - r) \geq \frac{3T}{2} + 1$ i.e. more than $\frac{3T}{2}$ blocks are mined in each chain. The same argument from the liveness proof for Π_{mod} now follows for why $\mathsf{Aequitas}(\cdot)$ will deliver tx. Consequently, Π_{multi} will satisfy $\frac{3T+2}{2g_0}$-liveness.

7 Applications

7.1 Zero-Block Confirmation

We now show that a fair ordering protocol can confirm non-conflicting transactions without the need to mine a single block. As mentioned in Section 1.1, this can be useful for preventing fraudulent

double-spending attempts. For this, we first show that any fair ordering protocols achieves a soft-ordering property: If a transaction tx is received at a honest node more than 2Δ time before tx', then tx is guaranteed to be ordered before tx' in the finalized ledger. We define this formally below.

Definition 7.1 (δ-Soft-Ordering)**.** We say that a protocol Π, satisfies δ-soft-confirmation (w.r.t. $(\mathcal{A}, \mathcal{Z})$) if the following property holds: Suppose that an honest node N receives tx in round r and tx' in a round greater than $r + \delta$. Then all honest nodes will output tx before tx' in the finalized log.

It is now easy to see that any protocol that satisfies receive-order-fairness also satisfies soft-ordering.

Theorem 7.2. *Suppose that* Π *satisfies* γ*-receive-order-fairness w.r.t.* $(\mathcal{A}, \mathcal{Z})$ *with network delay parameter* Δ*. Then,* Π *also satisfies* (2Δ)*-soft-ordering.*

Proof. The proof is straightforward. Suppose that a node N receives tx in round r. Then all honest nodes have received tx latest by round $r + \Delta$. Since node N received tx' at round later than $r + 2\Delta$, that implies that tx' was received at all other honest nodes earliest at round $r + \Delta + 1$.Therefore, (tx, tx') is such that all honest nodes received tx before they received tx', Since Π satisfies receive-order-fairness, tx will be ordered before tx' in the final output log by all honest nodes. Consequently, Π will satisfy (2Δ)-soft-ordering. Note that the same proof also works for fair-linearizability (from Section 2.3).

□

We now show that a protocol satisfying soft-ordering (for a small δ) can confirm non-conflicting transactions without the need to mine a single block. Define $\mathsf{conflict}(\text{tx}, \text{tx}') = \mathsf{conflict}(\text{tx}', \text{tx}) = 1$ if tx and tx' conflict with each other according to some semantic (e.g., spend the same tokens in a UTXO system). If two conflicting transactions are present in the output chain, only the first one will be executed by honest nodes to determine the current system state. Suppose that a node N receives tx in round r and is honest till at least round $r + \delta$. If N does not receive any tx' such that $\mathsf{conflict}(\text{tx}, \text{tx}') = 1$ till round $r + \delta$, then all honest nodes will output tx before any conflicting tx', and thus tx will be confirmed in the finalized ledger (due to the δ-soft-ordering property). This gives rise to a primitive that can be used to soft-confirm transactions. If δ is small, no blocks containing tx will be mined by this time, yet N will be able to soft-confirm tx, i.e., we get zero-block confirmation. Finally, we note that a transaction submitted by an honest node will not have any conflicting transactions at a future time, and thus can be confirmed by any other honest node within δ time. Thus protocols satisfying fair ordering can achieve confirmation of honest transactions in 2Δ time.

Remark. Although Π_{mod} and Π_{multi} satisfy block-order-fairness and not receive-order-fairness, we note that when $\Delta = 1$, they will satisfy (2Δ)-soft-ordering since any Condorcet cycle cannot extend past this time. In this time, even if no block containing the transaction tx has been mined, tx can still be locally confirmed, and consequently, we get zero-block-confirmation. Furthermore, the probability that a soft-confirmed transaction is not valid in the finalized ledger is the same as the probability that the protocol does not satisfy order-fairness, which is negligible in κ (and thus can be made arbitrarily small by increasing the number of chains as $g = \Theta(\kappa)$).

Soft-ordering is not a direct artifact of network synchrony. While soft-ordering is an easy corollary of order-fairness, we emphasize that is not directly a consequence of network synchrony in a permissionless setting. Even when transactions are gossiped and reach all nodes within a synchronous Δ period, a node will not be able to soft confirm within a small delay if the underlying consensus protocol is not fairly ordered. For instance, suppose that a node N received tx in round r and a conflicting transaction tx' in round $r + 2\Delta + 1$, i.e. all nodes receive tx before tx'. Still N cannot soft-confirm tx, since a miner proposing a block with tx' first may not be rejected by the network (since in the view of a honest node who receives tx in round $r + \Delta$ and tx' in round $r + \Delta + 1$, it is plausible that other nodes have received tx' first.)

Larger soft-confirm values (small enough to still be zero-block) do not work either. Even if it is common knowledge for the current nodes in the network that all nodes received tx before tx', the rejection of an adversarial miner's block with tx' first, will require this additional information. This means that the determination of whether a chain is valid is no longer only dependent on the transactions in it. Consequently, when new nodes enter the system, they will not be able to determine the correct honest chain to mine on.

Preventing DoS attacks. We assume a synchronous gossip-style network where transactions are gossiped and reach all nodes within a known Δ period. This may cause concerns of DoS attacks where an adversary strains the network by creating bogus or double-spend transactions. In the standard Bitcoin protocol, these transactions are not propagated by the network to avoid DoS problems. For our zero-block confirmation application, it is enough to only gossip the first instance of a double-spend for a token. The gossip of a single double-spend is enough for a node to know not to locally soft-confirm a transaction; other double-spends for the same token need not be gossiped to the entire network. Consequently, an adversary cannot cause a DoS attack on the network by creating bogus double-spend transactions. Note that this will not affect honest transactions that are not double-spent.

7.2 Decentralized Finance

The use of fair ordering protocols can greatly benefit decentralized finance applications. Consider automated market makers (AMMs) like Uniswap [4], Curve [5], etc., that allow users to exchange between tokens using an in-built price function to calculate the exchange rate. Here, the token exchange price that a user gets depends on the time her transaction executes. Using a fair ordering protocol can guarantee that users receive a fair price for their trades. Furthermore, it will also prevent miners from inserting their own transactions to take advantage of short-term changes in token exchange rates.

Order-fairness also grants fairness to applications whose incentives are based on earlier ordering. For example, in decentralized sealed-bid auctions, bids submitted before the auction close time cannot be forcefully rejected by a miner claiming to have received them later. Similarly, in an initial coin offering (ICO) token launch, if the developer wants to provide a cheaper rate for the first 1000 tokens, it will be able to do so in a fair "first come first serve" manner.

References

[1] `https://defipulse.com/`.

[2] https://www.coingecko.com/en/defi.

[3] https://medium.com/@amanusk/the-fastest-draw-on-the-blockchain-bzrx-example-6bd19fabdbe1.

[4] https://uniswap.org/.

[5] https://www.curve.fi/.

[6] Ittai Abraham et al. "Blinder – Scalable, Robust Anonymous Committed Broadcast". In: *CCS*. 2020, 1233–1252.

[7] Vivek Kumar Bagaria et al. "Prism: Deconstructing the Blockchain to Approach Physical Limits". In: *CCS*. 2019, pp. 585–602.

[8] Leemon Baird. *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*. https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf. 2016.

[9] Lorenz Breidenbach et al. *To Sink Frontrunners, Send in the Submarines*. https://hackingdistributed.com/2017/08/28/submarine-sends/.

[10] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *FOCS*. 2001, pp. 136–147.

[11] T-H. Hubert Chan et al. *Blockchain with Varying Number of Players*. Cryptology ePrint Archive, Report 2020/677. https://eprint.iacr.org/2020/677. 2020.

[12] *Condorcet Paradox*. https://wikipedia.org/wiki/Condorcet_paradox.

[13] Thomas Cormen et al. *Introduction to Algorithms*. 3rd. MIT Press, 2009.

[14] Phil. Daian et al. "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake". In: *FC*. 2019, pp. 23–41.

[15] Philip Daian et al. "Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability". In: *IEEE S&P*. 2020, pp. 585–602.

[16] Bernardo David et al. "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain". In: *EuroCrypt*. 2018, pp. 66–98.

[17] Shayan Eskandari et al. "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain". In: *FC*. 2019, pp. 170–189.

[18] Ittay Eyal and Emin Gün Sirer. "Majority is not Enough: Bitcoin Mining is Vulnerable". In: *FC*. 2014, pp. 436–454.

[19] Guy Fayolle et al. *Topics in the constructive theory of countable Markov chains*. Cambridge university press, 1995.

[20] Matthias Fitzi et al. *Ledger Combiners for Fast Settlement*. Cryptology ePrint Archive, Report 2020/675. https://ia.cr/2020/675. 2020.

[21] Juan Garay et al. "The Bitcoin Backbone Protocol with Chains of Variable Difficulty". In: *CRYPTO*. 2017, pp. 291–323.

[22] Juan A. Garay et al. "The Bitcoin Backbone Protocol: Analysis and Applications". In: *EUROCRYPT*. 2015, pp. 281–310.

[23] Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[24] Chi Ho et al. “Making distributed systems robust”. In: *OPODIS*. 2007, pp. 232–246.

[25] Ghassan O. Karame et al. “Double spending fast payments on Bitcoin”. In: *CCS*. 2012, pp. 906–917.

[26] Mahimna Kelkar et al. “Order-Fairness for Byzantine Consensus”. In: *CRYPTO*. 2020, pp. 451–480.

[27] Aggelos Kiayias et al. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”. In: *CRYPTO*. 2017, pp. 357–388.

[28] Klaus Kursawe. *Wendy, the Good Little Fairness Widget*. Cryptology ePrint Archive, Report 2020/885. `https://ia.cr/2020/885`. 2020.

[29] Michael Lewis. *Flash Boys: A Wall Street Revolt*. WW Norton & Company, 2014.

[30] Chenxin Li et al. “A decentralized blockchain with high throughput and fast confirmation”. In: *ATC*. 2020, pp. 515–528.

[31] Andrew Miller et al. “The Honey Badger of BFT Protocols”. In: *ACM CCS*. 2016, pp. 31–42.

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

[33] Rafael Pass et al. “Analysis of the Blockchain Protocol in Asynchronous Networks”. In: *EUROCRYPT*. 2017, pp. 643–673.

[34] Rafael Pass and Elaine Shi. “FruitChains: A Fair Blockchain”. In: *PODC*. 2017, pp. 315–324.

[35] Rafael Pass and Elaine Shi. “Hybrid Consensus: Efficient Consensus in the Permissionless Model”. In: *DISC*. Vol. 91. 2017, 39:1–39:16.

[36] Rafael Pass and Elaine Shi. “Rethinking Large-Scale Consensus”. In: *CSF*. 2017, pp. 115–129.

[37] Rafael Pass and Elaine Shi. “Thunderella: Blockchains with Optimistic Instant Confirmation”. In: *EUROCRYPT*. 2018, pp. 3–33.

[38] Ling Ren and Srinivas Devadas. “Proof of space from stacked expanders”. In: *TCC*. 2016, pp. 262–285.

[39] Walter Rudin et al. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York, 1964.

[40] *Smith Criterion*. `https://en.wikipedia.org/wiki/Smith_criterion`.

[41] Haifeng Yu et al. “Ohie: Blockchain scaling made simple”. In: *IEEE S&P*. 2020, pp. 90–105.

[42] Yunhao Zhang et al. “Byzantine Ordered Consensus without Byzantine Oligarchy”. In: *OSDI*. 2020, pp. 633–649.

A Cross-chain-quality proof for Π_{multi}

We restate the cross-chain-quality property below. We will show that it holds, except for a negligible number of views.

Property A.1 (Cross-Chain-Quality)**.** For a given view and tx, let $\mathcal{B}_{\text{tx}}$ be the set of blocks that contain tx for the first time in the confirmed part (i.e. $\mathsf{chain}_j[:-T]$) of each of the g chains. Then, more than half of the blocks in $\mathcal{B}_{\text{tx}}$ were mined by honest nodes.

Notation. Before proceeding with the proof, we start by introducing some useful notation from [7]. Let $f = npq$ denote the mining rate (as a function of κ) per round for one chain. Typically, in practice, the hardness parameter p is set as $p = \Theta(\frac{1}{\Delta qn})$ which would make $f = \Theta(1)$ in κ since we are using $\Delta = 1$. This will make our analysis cleaner. Recall that g is the number of parallel chains and β is the adversarial corruption threshold. We use the following random variables in our proofs. For a chain i, and round r, we define:

- $X_i[r] = \begin{cases} 0, & \text{if no honest node mines a block in round } r \\ 1, & \text{if at least one honest node mines a block in round } r \end{cases}$
- $X_i[r_1, r_2] = \sum_{r=r_1}^{r_2} X_i[r]$.
- $Y_i[r] = \begin{cases} 1, & \text{if exactly one honest node mines a block in round } r \\ 0, & \text{otherwise} \end{cases}$
- $Y_i[r_1, r_2] = \sum_{r=r_1}^{r_2} Y_i[r]$.
- $Z_i[r]$ is the number of blocks mined by an adversarial node in round r.
- $Z_i[r_1, r_2] = \sum_{r=r_1}^{r_2} Z_i[r]$.
- $C[:\ell]$ is the prefix of the longest chain C until the level ℓ.
- $C[\ell]$ is the block at level ℓ of the chain C.

For an event Q, use $\mathbf{1}_Q$ to denote the indicator variable for when Q holds. That is, $\mathbf{1}_Q = 1$ when Q is true and 0 otherwise. We use the notation $a_+ = \max\{0, a\}$.

Following [7], $Z_i[r]$ follows a Poisson distribution, while $X_i[r]$ follows a Bernoulli distribution. Specifically, we have,

- $Z_i[r] \sim \text{Poisson}(\beta f)$
- $X_i[r] \sim \text{Bernoulli}(1 - e^{-(1-\beta)f})$

A.1 Proof structure.

In order to prove the cross-chain-quality property for Π_{multi}, abstractly we will construct an event $V_i[r]$ for each chain i at round r, which will guarantee that any transaction first seen in round r will be contained in a block mined by an honest node. Now, the probability in an arbitrary chain i at any round r, that $V_i[r]$ happens (as a function of β and f) is larger than 0.5, then we can set the number of chains $g = \Theta(\kappa)$, and use the Chernoff bound to argue that the probability that any transaction is mined in more than half the parallel chains by the adversary is negligible in κ. We elaborate on the main proof parts below.

Constructing $V_i[r]$. We start by defining the following random processes for all chains i and rounds $r > 0$.

$$S_i[r] = (S_i[r-1] + Z_i[r] - X_i[r])_+$$
$$L_i[r,t] = X_i[r,t] - Z_i[r,t].$$

We also set $S_i[0] = 0$. Now, if the adversary never reveals the blocks it mines, $S_i[r]$ captures the difference between the length of longest private adversarial chain i at round r, and the longest public chain held by honest nodes. The random process $L_i[r,t]$ represents the difference between the number of honest blocks and adversarial blocks mined in chain i from round r to round t.

Now, we define $V_i[r]$ for chain i at round $r > 0$ as the event that all of the following holds:

1. $S_i[r-1] = 0$
2. There exists a round $t' > r$ such that
   - $Z_i[r,t'] = 0$. In other words, the adversary was not able to mine a block from round r to t'.
   - $X_i[r,t'-1] = 0$. In other words, honest nodes were not able to mine any block from r until $t'-1$.
   - $Y_i[t'] = 1$. In other words, exactly one honest node mined a block in round t'.
   - $L_i[r+1,t] \geq 1 \quad \forall t > t'$. In other words, the number of honest blocks mined in chain i from round $r+1$ until round t is greater than the corresponding number of adversarial blocks in the same interval, for all $t > t'$.

We will call the first event $V_i^1[r]$ and the second event $V_i^2[r]$. We evaluate their probabilities separately since they are independent.

Honest block when $V_i[r]$ holds. In Section A.2, we show that whenever $V_i[r]$ holds, any transaction tx that is first revealed (to honest nodes) at round r, in chain i, would be contained in a block mined by honest miner. Intuitively, there are two ways a transaction can be propagated:

- *Honestly propagated transaction*: A transaction that is input to all nodes (honest and adversarial) within $\Delta = 1$ rounds.

- *Adversarially propagated transaction*: A transaction that is created by the adversary and is first seen by an honest node only through the mining of a PoW block by the adversary. Here, the adversary knows the transaction earlier than honest nodes and can get a head-start to mine it into the chains.

Our proof of the result holds regardless of how the transaction is propagated.

Probability of $V_i[r]$. In Section A.3, we compute the probability, as a function of β and f, that the event $V_i[r]$ occurs for a given round r. Denote this by $\phi(\beta, f)$. Next, we find the threshold β_{thresh} such that for $\beta < \beta_{\text{thresh}}$, we have $\phi(\beta, f) > 0.5$. This means that for a transaction first revealed in round r, the probability for each chain that tx is contained in a block mined by an honest node is more than 0.5.

Achieving cross-chain-quality. We can now leverage the Chernoff bound to get a lower bound on the number of chains g, so that the probability that the adversary mined tx into more than half of the parallel chains (or equivalently, cross-chain-quality is not satisfied) is negligible in the security parameter. We show this in Section A.4.

A.2 Honest block under $V_i[r]$

In this section, we will show that $V_i[r]$ guarantees that any transaction (honest or adversarial) first input to honest nodes at round r will be contained in an honest block in chain i.

For this, we need to show that the following hold, except with negligible probability.

- At round r, the adversary does not have a longer private adversarial chain for chain i than the longest honest chain (Lemma A.2).
- The first block to be mined in chain i after round r is an honest block and this honest block persists in the longest chain for all future rounds (Lemma A.3).

Lemma A.2. *Consider a round when $S_i[r] = 0$. The adversary cannot posses a longer chain than the chain held by the honest users at round r.*

Proof. The proof proceeds by contradiction. Suppose r was a round with $S_i[r] = 0$ and the adversary possessed a longer chain C_i^a. Let the level of the tip of the longest chain held by the honest users, C_i^h, at round r be ℓ_{tip}. Let,

- $\ell_{common} = \max\{\ell : C_i^a[:\ell] = C_i^h[:\ell]\}$ be the last common ancestor,
- $\ell' = \max\{\ell \leq \ell_{common}$ such that $C_i^h[\ell]$ is honest block$\}$ and r' be the round when it was mined.

Observe that for each of the blocks in the chain C_i^h from level $\ell_{common} + 1$ to level ℓ_{tip}, there is a corresponding adversarial block in the adversarial chain C_i^a. Also, if $\ell' < \ell_{common}$, then all the blocks from levels $\ell' + 1$ to ℓ_{common} in the chain C_i^h are adversarial blocks. Thus, from $r' + 1$ to

round r, the number of adversarial blocks mined is greater than the number of rounds where at least one honest block was mined. That is,

$$Z_i[r'+1, r] \geq X_i[r'+1, r] + 1$$

However, we also have $S_i[r'] \geq 0 = S_i[r]$, and so,

$$\begin{aligned} 0 &\geq S_i[r] - S_i[r'] \\ &\geq \sum_{s=r'+1}^{r} Z_i[s] - X_i[s] \\ &= Z_i[r'+1, r] - X_i[r'+1, r] \end{aligned}$$

i.e. $Z_i[r'+1, r] \leq X_i[r'+1, r]$ which is a contradiction. □

Lemma A.3. *Suppose $V_i[r]$ happens at round r. Then, the first block B mined at round $t' > r$ is honest and will persist in the longest chain.*

Proof. First, observe that since $Z_i[r, t'] = 0$, we have $S_i[r-1] = S_i[r] = \cdots = S_i[t'] = 0$. Now, by Lemma A.2, the adversary doesn't possess a longer chain in the rounds $r-1, r, \cdots, t'$. Also, $Z_i[r, t'] = 0$ implies that no new adversarial block was mined in in the rounds $r-1, r, \cdots, t'$ and exactly one honest node mines a block in round t'. So, the chain i held by honest users at round t' is longer than any adversarial chain and thus, contains the block B.

Next, by contradiction, let $r^* > t'$ be the earliest time when the honest users hold a chain that contains a different block at the same level where B was. Thus, at round r^*, there exists an adversarial chain that is at least as long the honest chain containing the block B. Let round r' be as defined in Lemma A.2. Then, reasoning as in Lemma A.2, from round $r'+1$ to round r^*, the number of adversarial blocks mined is greater than or equal to the number of rounds where at least one honest block is mined. That is,

$$Z_i[r'+1, r^*] \geq X_i[r'+1, r^*]$$

Also, we have

$$\begin{aligned} & S_i[r'] \geq 0 = S_i[r] \\ \implies\ & Z_i[r'+1, r] \leq X_i[r'+1, r] \end{aligned}$$

This implies that $Z_i[r+1, r^*] \geq X_i[r+1, r^*]$. However, as event $V_i[r]$ occurs at round r, so, we have

$$\begin{aligned} L_i[r+1, r^*] &\geq 1 \\ X_i[r+1, r^*] - Z_i[r+1, r^*] &\geq 1 \\ X_i[r+1, r^*] &> Z_i[r+1, r^*] \end{aligned}$$

which leads to contradiction. □

A.3 Computation of $\phi(f, \beta)$

In this section, we compute the probability $\phi(f, \beta)$, of the event $V_i[r]$. For this, we will prove the following lemma.

Lemma A.4.

$$\lim_{f \to 0^+} \phi(f, \beta) = \frac{(1-2\beta)^2}{1-\beta}.$$

We split this computation into two parts. We compute the probability of the event $V_i^1[r]$ in Section A.3.1, and the probability of $V_i^2[r]$ given $V_i^1[r]$ in Section A.3.2. Finally, we can combine the two probabilities to compute $\phi(f, \beta)$.

A.3.1 Probability of $V_i^1[r]$

Recall that $V_i^1[r]$ is the event that $S_i[r-1] = 0$. To compute the probability of this event, we will compute the stationary distribution of the discrete time Markov chain $\{S_i[r]\}_{r \geq 0}$.

First, for a chain i, and round r, we define the following random variable:

$$\begin{aligned} S_i[r] &= (S_i[r-1] + Z_i[r] - X_i[r])_+ \\ S_i[r] &= 0 \end{aligned}$$

Observe that due to the memory-less property of both honest and adversarial mining (since mining is through a random oracle), for all $r > 0$, we have,

$$P(S_i[r] = s \mid S_i[r-1] = s_{r-1}, \cdots, S_i[0] = 0) = P(S_i[r] = s \mid S_i[r-1] = s_{r-1})$$

Therefore, the process $\{S_i[r]\}_{r \geq 0}$ gives rise to a discrete time Markov chain (DTMC). We now show that $\{S_i[r]\}_{r \geq 0}$ is positive recurrent in the following lemma.

Lemma A.5. *If $\mathbb{E}[Z_i[r]] < \mathbb{E}[Y_i[r]]$, then, $\{S_i[r]\}_{r \geq 0}$ is positive recurrent.*

Proof. First, notice from the definition of $S_i[r]$ that

$$\begin{aligned} (S_i[r+1])^2 &\leq (S_i[r] + Z_i[r+1] - Y_i[r+1])^2 \\ &= (Z_i[r+1] - Y_i[r+1])^2 + (S_i[r])^2 + 2S_i[r](Z_i[r+1] - Y_i[r+1]) \end{aligned}$$

Now, for each chain i, define a drift function as:

$$\mathfrak{d}S_i[r] = \frac{1}{2}(S_i[r+1])^2 - \frac{1}{2}(S_i[r])^2$$

Then,

$$\begin{aligned} \mathfrak{d}S_i[r] &\leq \frac{1}{2}(Z_i[r+1] - Y_i[r+1])^2 + S_i[r]\,(Z_i[r+1] - X_i[r+1]) \\ \mathbb{E}\left[\mathfrak{d}S_i[r] \mid S_i[r]\right] &\leq \left(\frac{1}{2}\mathbb{E}\left[(Z_i[r+1] - X_i[r+1])^2 \mid S_i[r]\right]\right) \\ &\quad + (S_i[r]\mathbb{E}\left[(Z_i[r+1] - X_i[r+1]) \mid S_i[r]\right)] \end{aligned}$$

Recall that $Z_i[r+1] \sim \text{Poisson}(\beta f)$ and $X_i[r+1] \sim \text{Bernoulli}(1-e^{-(1-\beta)f})$ and so, $Z_i[r+1]$ and $X_i[r+1]$ have bounded second moments. Therefore, the first term in the expression before can be bounded as $\frac{1}{2}\mathbb{E}\left[(Z_i[r+1]-X_i[r+1])^2 \mid S_i[r]\right] \leq B$, where $B = B(f,\beta)$ depends on f and β. Thus,

$$\begin{aligned}\mathbb{E}\left[\mathfrak{d}S_i[r] \mid S_i[r]\right] &\leq B + S_i[r]\mathbb{E}\left[(Z_i[r+1]-X_i[r+1]) \mid S_i[r]\right] \\ &= B + S_i[r]\left(\mathbb{E}[Z_i[r+1]] - \mathbb{E}[X_i[r+1]]\right)\end{aligned}$$

However, $\mathbb{E}[Z_i[r+1]] - \mathbb{E}[X_i[r+1]] < -\epsilon$, for some $\epsilon > 0$. Therefore, we have

$$\mathbb{E}\left[\mathfrak{d}S_i[r] \mid S_i[r]\right] \leq B - \epsilon S_i[r]$$

To summarize,

- $\mathbb{E}\left[\mathfrak{d}S_i[r] \mid S_i[r]\right] \leq -\epsilon^*$ for some $\epsilon^* > 0$, if $S_i[r] > \frac{B}{\epsilon}$
- $\mathbb{E}\left[\mathfrak{d}S_i[r] \mid S_i[r]\right] \leq B$ otherwise.

Thus, by Foster's Theorem [19], we conclude that $\{S_i[r]\}_{r\geq 0}$ is positive recurrent. □

Now, $\{S_i[r]\}_{r\geq 0}$ is irreducible since it is possible to get from any state to any other state (either by honest block mining or by adversarial block mining). Furthermore, it is aperiodic since it has self-loops, and therefore a 1-step transition can retain the earlier state. Therefore the stationary distribution exists. If $\boldsymbol{\pi}$ is the stationary distribution of $\{S_i[r]\}_{r\geq 0}$, then, by ergodic theorem [23], we have

$$\lim_{r\to\infty} P(S_i[r]=0) = \pi_0$$

where π_0 denotes the stationary distribution probability at state 0.

We will now calculate the probability of the stationary distribution π_0 under a general f. For this, we will write the transition probability matrix for the DTMC $\{S_i[r]\}_{r\geq 0}$. Define $\mathtt{p}_X^0 = P(X_i[r]=0) = e^{-(1-\beta)f}$, $\mathtt{p}_X^1 = P(X_i[r]=1) = 1-e^{-(1-\beta)f}$, and $\mathtt{p}_Z^k = P(Z_i[r]=k) = \frac{(\beta f)^k e^{-\beta f}}{k!}$. Then, the transition probability matrix is given by

$$\begin{pmatrix}
\mathtt{p}_Z^0+\mathtt{p}_Z^1\mathtt{p}_X^1 & \mathtt{p}_Z^1\mathtt{p}_X^0+\mathtt{p}_Z^2\mathtt{p}_X^1 & \mathtt{p}_Z^2\mathtt{p}_X^0+\mathtt{p}_Z^3\mathtt{p}_X^1 & \mathtt{p}_Z^3\mathtt{p}_X^0+\mathtt{p}_Z^4\mathtt{p}_X^1 & \cdots \\
\mathtt{p}_Z^0\mathtt{p}_X^1 & \mathtt{p}_Z^0\mathtt{p}_X^0+\mathtt{p}_Z^1\mathtt{p}_X^1 & \mathtt{p}_Z^1\mathtt{p}_X^0+\mathtt{p}_Z^2\mathtt{p}_X^1 & \mathtt{p}_Z^2\mathtt{p}_X^0+\mathtt{p}_Z^3\mathtt{p}_X^1 & \cdots \\
0 & \mathtt{p}_Z^0\mathtt{p}_X^1 & \mathtt{p}_Z^0\mathtt{p}_X^0+\mathtt{p}_Z^1\mathtt{p}_X^1 & \mathtt{p}_Z^1\mathtt{p}_X^0+\mathtt{p}_Z^2\mathtt{p}_X^1 & \cdots \\
0 & 0 & \mathtt{p}_Z^0\mathtt{p}_X^1 & \mathtt{p}_Z^0\mathtt{p}_X^0+\mathtt{p}_Z^1\mathtt{p}_X^1 & \cdots \\
0 & 0 & 0 & \mathtt{p}_Z^0\mathtt{p}_X^1 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

The important observation is that this matrix is of the form

$$\begin{pmatrix} a & b_2 & b_3 & \cdots \\ b_0 & b_1 & b_2 & \cdots \\ 0 & b_0 & b_1 & \cdots \\ 0 & 0 & b_0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

with $a = b_0 + b_1$. Therefore, we get the following relationship between the stationary probabilities

$$\begin{aligned} \pi_0 &= a\pi_0 + b_0\pi_1 \\ \pi_j &= \sum_{k=0}^{j+1} b_{i+1-k}\pi_k \qquad\qquad (j > 1) \end{aligned}$$

Rearranging the terms of the second equation, we get

$$\begin{aligned} (1 - b_1)\pi_j &= b_0\pi_{j+1} + \sum_{k=0}^{j-1} b_{j+1-k}\pi_k \\ \therefore (1 - b_1)\pi_j z^{-j} &= b_0\pi_{j+1}z^{-j} + \sum_{k=0}^{j-1} b_{j+1-k}\pi_k z^{-j} \\ \therefore (1 - b_1)\sum_{j\geq 1}\pi_j z^{-j} &= b_0\sum_{j\geq 1}\pi_{j+1}z^{-j} + \sum_{j\geq 1}\sum_{k=0}^{j-1} b_{j+1-k}\pi_k z^{-j} \end{aligned}$$

Notice that the summation $\sum_{j\geq 1}\sum_{k=0}^{j-1}$ can be written in terms of $\sum_{k\geq 0}\sum_{j\geq k+1}$. Now, define the unilateral z-transforms $\mathfrak{P}(z) = \sum_{j\geq 0}\pi_j z^{-j}$ and $\mathfrak{G}(z) = \sum_{j\geq 0} b_j z^{-j}$. See [23] for a primer on the z-transform. Then,

$$\begin{aligned} (1 - b_1)(\mathfrak{P}(z) - \pi_0) &= b_0 z\left(\mathfrak{P}(z) - \pi_0 - \frac{\pi_1}{z}\right) + \sum_{k\geq 0}\sum_{j\geq k+1} b_{j+1-k}\pi_k z^{-j} \\ &= b_0 z\left(\mathfrak{P}(z) - \pi_0 - \frac{\pi_1}{z}\right) + z\sum_{k\geq 0}\pi_k z^{-k}\sum_{j\geq k+1} b_{j+1-k}z^{-(j+1-k)} \\ &= b_0 z\left(\mathfrak{P}(z) - \pi_0 - \frac{\pi_1}{z}\right) + z\mathfrak{P}(z)\left(\mathfrak{G}(z) - b_0 - \frac{b_1}{z}\right) \end{aligned}$$

Rearranging, we get

$$\begin{aligned} \mathfrak{P}(z) &= \frac{\pi_0(1 - b_1) - b_0 z(\pi_0 + \frac{\pi_1}{z})}{1 - b_0 z - z\mathfrak{G}(z) + b_0 z} \\ &= \frac{\frac{\pi_0}{z}(1 - b_1) - b_0(\pi_0 + \frac{\pi_1}{z})}{\frac{1}{z} - b_0 - \mathfrak{G}(z) + b_0} \end{aligned}$$

Notice that the boundary condition is at $z = 1$, that is, $\mathfrak{P}(1) = 1$. Therefore, applying L'Hôpital's rule, we get,

$$\lim_{z\to 1} P(z) = \lim_{z\to 1} \frac{\pi_0(1-b_1) - b_0\pi_1}{1-\mathfrak{G}'(z)} = \frac{\pi_0(1-b_1) - b_0\pi_1}{1-\mathfrak{G}'(1)}$$

where $\mathfrak{G}'$ is the derivative of $\mathfrak{G}$ with respect to $\frac{1}{z}$. Consequently,

$$\begin{aligned}\pi_0(1-b_1) - (1-a)\pi_0 &= 1-\mathfrak{G}'(1)\\ \therefore \pi_0 &= \frac{1-\mathfrak{G}'(1)}{a-b_1}\\ \therefore \pi_0 &= \frac{1-\mathfrak{G}'(1)}{b_0}\end{aligned}$$

where in the last step we used the fact that $a = b_0 + b_1$. Next, we will evaluate $\mathfrak{G}'(1)$. We have

$$\begin{aligned}\mathfrak{G}(z) &= \sum_{j\geq 0} b_j z^{-j}\\ \mathfrak{G}'(z) &= \sum_{j\geq 1} j b_j z^{-(j-1)}\end{aligned}$$

where we differentiated $\mathfrak{G}(z)$ with respect to $\frac{1}{z}$. Thus, we have

$$\begin{aligned}\mathfrak{G}'(1) &= \sum_{j\geq 1} j b_j\\ &= \sum_{j\geq 1} j\left[\mathrm{p}_Z^{j-1}\mathrm{p}_X^0 + \mathrm{p}_Z^j\mathrm{p}_X^1\right]\\ &= \sum_{j\geq 1} j\left[\frac{(\beta f)^{j-1}e^{-\beta f}}{(j-1)!}e^{-(1-\beta)f} + \frac{(\beta f)^j e^{-\beta f}}{j!}\left[1-e^{-(1-\beta)f}\right]\right]\\ &= e^{-f}\sum_{j\geq 1} j\frac{(\beta f)^{j-1}}{(j-1)!} + \beta f e^{-\beta f}\left[1-e^{-(1-\beta)f}\right]\sum_{j\geq 1}\frac{(\beta f)^{j-1}}{(j-1)!}\\ &\overset{(a)}{=} e^{-f}\left[e^{\beta f} + \beta f e^{\beta f}\right] + \beta f e^{-\beta f}\left[1-e^{-(1-\beta)f}\right]e^{\beta f}\\ &= \beta f + e^{-(1-\beta)f}\end{aligned}$$

where in (a), we used the fact that $xe^x + x = \sum_{j\geq 1} j\frac{x^{j-1}}{(j-1)!}$ and $e^x = \sum_{j\geq 0}\frac{x^j}{j!}$. Consequently,

$$\pi_0 = \frac{1-\beta f - e^{-(1-\beta)f}}{e^{-\beta f}\left[1-e^{-(1-\beta)f}\right]}$$

Now, for $f \to 0^+$, L'Hôpital's rule can be used to evaluate π_0, which gives $\pi_0 = \frac{1-2\beta}{1-\beta}$. Therefore, $\lim_{r\to\infty} P(S_i[r] = 0) = \frac{1-2\beta}{1-\beta}$ as $f \to 0^+$. This means that for sufficiently small f, the probability will be sufficiently close to $\frac{1-2\beta}{1-\beta}$. Furthermore, although the stationary distribution is defined as the round number $r \to \infty$, it will converge exponentially towards the stationary probability (see [23] for more details). This means that for a given f, at any round r after a warmup time of $T_{\text{warmup}} = \Theta(\kappa)$, the probability will be close to the stationary probability, and we can replace it with the stationary probability henceforth to simplify our computation.

A.3.2 Probability of $V_i^2[r]$

Next, we compute the probability of the event $V_i^2[r]$. Recall that $V_i^2[r]$ is the event as follows:

$$V_i^2[r] = \left\{ \exists t' > r \quad s.t. \quad \begin{smallmatrix} Z_i[r,t']=0 \\ X_i[r,t'-1]=0 \\ Y_i[t']=1 \\ L_i[r+1,t] \geq 1 \quad \forall t > t' \end{smallmatrix} \right\}$$

First, for a chain i and round $k > 0$, we define the following random walk:

$$\begin{aligned} D_i[k] &= D_i[k-1] + X_i[k] - Z_i[k] \\ D_i[0] &= 1 \end{aligned}$$

This random walk counts the difference between the number of honest blocks and number of adversarial blocks mined from round 1 to round k given that the random walk started from the initial state of $D_i[0] = 1$. Suppose that we start the random walk $\{D_i[k]\}_{k \geq 0}$ at round t'. Then,

- $Z_i[r, t'] = 0, X_i[r, t' - 1] = 0$ and $Y_i[t'] = 1$ implies that two honest blocks and no adversarial blocks were mined from the round r to round t'. Therefore, we set $D_i[0] = 1$.
- For any $k > 0$, $D_i[k] = L[r + 1, t' + k]$.

Now, $V_i^2[r] = \{D_i[0] = 1, D_i[k] \geq 1 \quad \forall k > 0\}$, and therefore, we get:

$$\begin{aligned} P(V_i^2[r]) &= \{D_i[0] = 1, D_i[k] \geq 1 \quad \forall k > 0\} \\ &= P(D_i[0] = 1) \times P\left(D_i[k] \geq 1 \quad \forall k > 0 \mid D_i[0] = 1\right) \end{aligned}$$

First, we compute $P(D_i[0] = 1)$ as

$$\begin{aligned} P(D_i[0] = 1) &= P\left(\begin{smallmatrix} \exists t' > r \text{ such that} \\ Z_i[r,t']=0 \\ X_i[r,t'-1]=0 \\ Y_i[t']=1 \end{smallmatrix} \right) \\ &= \sum_{m=1}^{\infty} P\left(\begin{smallmatrix} t'=r+m \\ Z_i[r,t']=0 \\ X_i[r,t'-1]=0 \\ Y_i[t']=1 \end{smallmatrix} \right) \\ &= \sum_{m=1}^{\infty} \left(e^{-(1-\beta)f}\right)^m \left(e^{-\beta f}\right)^{m+1} (1-\beta) f e^{-(1-\beta)f} \\ &= \frac{(1-\beta) f e^{-2f}}{1 - e^{-f}} \end{aligned}$$

Now, we can use L'Hôpital's rule, to get $\lim_{f \to 0^+} P(D_i[0] = 1) = 1 - \beta$.

In order to compute $P\left(D_i[k] \geq 1 \quad \forall k > 0 \mid D_i[0] = 1\right)$, we first define

$$\mathbf{P}_u = P\left(D_i[r] \geq 1 \quad \forall r \geq 0 \mid D_i[0] = u\right)$$

Observe that the boundary conditions are $\mathbf{P}_0 = 0$ and $\mathbf{P}_\infty = 1$. Now for $u \geq 1$,

$$
\begin{aligned}
\mathbf{P}_n &= P\left(D_i[r] \geq 1 \quad \forall r \geq 0 \mid D_i[0] = u\right) \\
&= \sum_{k=-1}^{\infty} P\left(X_i[1] - Z_i[1] = -k, D_i[r] \geq 1 \quad \forall r \geq 0 \mid D_i[0] = u\right) \\
&= \sum_{k=-1}^{\infty} P\left(X_i[1] - Z_i[1] = -k\right) P\left(D_i[r] \geq 1 \quad \forall r \geq 0 \mid D_i[1] = u - k\right) \\
&= \sum_{k=-1}^{u-1} P\left(X_i[1] - Z_i[1] = -k\right) P\left(D_i[r] \geq 1 \quad \forall r \geq 0 \mid D_i[1] = u - k\right) \\
&= \sum_{k=-1}^{u-1} P\left(X_i[1] - Z_i[1] = -k\right) \mathbf{P}_{u-k} \\
&= a\mathbf{P}_{u+1} + c\mathbf{P}_u + \sum_{k=1}^{u-1} b_k \mathbf{P}_{u-k}
\end{aligned}
$$

where $a = P(X_i[1] - Z_i[1] = 1)$, $c = P(X_i[1] - Z_i[1] = 0)$ and $b_k = P(X_i[1] - Z_i[1] = -k)$. Therefore,

$$
(1-c)\mathbf{P}_u z^{-u} = a\mathbf{P}_{u+1} z^{-u} + \sum_{k=1}^{u-1} b_k \mathbf{P}_{u-k} z^{-u}
$$

$$
(1-c)\sum_{u\geq 1} \mathbf{P}_u z^{-u} = az \sum_{u \geq 1} \mathbf{P}_{u+1} z^{-(u+1)} + \sum_{u \geq 1} \sum_{k=1}^{u-1} b_k \mathbf{P}_{u-k} z^{-u}
$$

We now define the z-transforms $\mathfrak{P}(z) = \sum_{u\geq 1} \mathbf{P}_u z^{-u}$ and $\mathfrak{B}(z) = \sum_{k \geq 0} b_{k+1} z^{-k}$. We can skip $\mathbf{P}_0$ as $\mathbf{P}_0 = 0$. Therefore,

$$
\begin{aligned}
(1-c)\mathfrak{P}(z) &= az(\mathfrak{P}(z) - \mathbf{P}_1 z^{-1}) + \sum_{k\geq 1} \sum_{u \geq k+1} b_k \mathbf{P}_{u-k} z^{-k} z^{-(u-k)} \\
&= a(z\mathfrak{P}(z) - \mathbf{P}_1) + \frac{1}{z} \sum_{k \geq 1} b_k z^{-(k-1)} \sum_{u \geq k+1} P_{u-k} z^{-(u-k)} \\
&= a(z\mathfrak{P}(z) - \mathbf{P}_1) + \frac{1}{z} \mathfrak{B}(z)\mathfrak{P}(z) \\
\therefore \mathfrak{P}(z) &= \frac{a\mathbf{P}_1 z}{az^2 - (1-c)z + \mathfrak{B}(z)}
\end{aligned}
$$

Recall that,

- $a = P(X_i[1] - Z_i[1] = 1) = P(X_i[1] = 1)P(Z_i[1] = 0) = (1 - e^{-(1-\beta)f})e^{-\beta f} = e^{-\beta f} - e^{-f}$
- $c = P(X_i[1] - Z_i[1] = 0)$
  $= P(X_i[1] = 0)P(Z_i[1] = 0) + P(X_i[1] = 1)P(Z_i[1] = 1)$
  $= e^{-(1-\beta)f} e^{-\beta f} + (1 - e^{-(1-\beta)f})\beta f e^{-\beta f}$
  $= e^{-f} + \beta f(e^{-\beta f} - e^{-f})$

- $b_k = P(X_i[r] - Z_i[r] = -k)$
  $= P(X_i[r] = 0)P(Z_i[r] = k) + P(X_i[r] = 1)P(Z_i[r] = k+1)$
  $= e^{-(1-\beta)f}\frac{(\beta f)^k e^{-\beta f}}{k!} + (1 - e^{-(1-\beta)f})\frac{(\beta f)^{k+1}e^{-\beta f}}{(k+1)!}$
  $= e^{-f}\frac{(\beta f)^k}{k!} + (e^{-\beta f} - e^{-f})\frac{(\beta f)^{k+1}}{(k+1)!}$

Therefore, $\lim_{f\to 0^+} \mathfrak{B}(z) = 0$. Now, by L'Hôpital's rule,

$$\begin{aligned}
\lim_{f\to 0^+} \mathfrak{P}(z) &= \frac{(1-\beta)\mathbf{P}_1 z}{(1-\beta)z^2 - z + \beta} \\
&= \frac{(1-\beta)\mathbf{P}_1}{\beta}\left[\left(\frac{1-\beta}{1-2\beta}\right)\left(\frac{1}{z^{-1} - \frac{1-\beta}{\beta}}\right) - \left(\frac{\beta}{1-2\beta}\right)\left(\frac{1}{z^{-1}-1}\right)\right] \\
&= \frac{(1-\beta)\mathbf{P}_1}{1-2\beta}\left[\left(\frac{1-\beta}{\beta}\right)\left(\frac{1}{z^{-1} - \frac{1-\beta}{\beta}}\right) - \left(\frac{1}{z^{-1}-1}\right)\right] \qquad (1)
\end{aligned}$$

To determine $\mathbf{P}_u$, we need to perform an inverse z-transform of $\mathfrak{P}(z)$ as shown below:

$$\mathbf{P}_u = \frac{1}{2\pi j}\oint_C \mathfrak{P}(z) z^{u-1} dz$$

where j is the imaginary unit, and C is a counterclockwise closed path encircling the origin and entirely lying in the region of convergence. Next, we want to determine $\lim_{f\to 0^+}\lim_{n\to\infty}\mathbf{P}_u$. However, we have,

$$\lim_{f\to 0^+}\lim_{u\to\infty}\mathbf{P}_u = \lim_{u\to\infty}\lim_{f\to 0^+}\mathbf{P}_u$$

wherein we used Moore-Osgood theorem [39] because:

- $\lim_{n\to\infty}\mathbf{P}_u$ exists pointwise for each different $f \neq 0$ because, by definition, $0 \leq \mathbf{P}_u \leq 1$ for all $u \geq 0$
- $\lim_{f\to 0^+}\mathbf{P}_u$ converges uniformly. This is because,
  1. The integral in the inverse z-transform is over a contour C that lies in the region of convergence and so $\mathfrak{P}(z)z^{u-1}$ is dominated by some Lebesgue integral function $\mathfrak{S}(z)$ in the sense that $|\,\mathfrak{P}(z)z^{u-1}\,| \leq \mathfrak{S}(z)$.
  2. Therefore, by the dominated convergence theorem [39] and using equation 1, we can interchange the limit and the contour integral to obtain

$$\begin{aligned}
\lim_{f\to 0^+}\mathbf{P}_u &= \lim_{f\to 0^+}\frac{1}{2\pi j}\oint_C \mathfrak{P}(z)z^{u-1}dz = \frac{1}{2\pi j}\oint_C \lim_{f\to 0^+}\mathfrak{P}(z)z^{u-1}dz \\
&= \mathbf{P}_1\left(\frac{1-\beta}{1-2\beta}\right)\left[1 - \left(\frac{\beta}{1-\beta}\right)^u\right].
\end{aligned}$$

  Further, note that $0 \leq \mathbf{P}_1 \leq 1$.
  3. Observe that $\mathbf{P}_1\left(\frac{1-\beta}{1-2\beta}\right)\left[1 - \left(\frac{\beta}{1-\beta}\right)^u\right]$ uniformly converges to $\mathbf{P}_1\left(\frac{1-\beta}{1-2\beta}\right)$.

Recall now, that our boundary condition was $P_\infty = 1$ i.e.

$$1 = \lim_{u\to\infty} \mathbf{P}_u = \lim_{f\to 0^+} \lim_{u\to\infty} \mathbf{P}_u = \mathbf{P}_1 \left(\frac{1-\beta}{1-2\beta}\right)$$

Finally, we get

$$\mathbf{P}_1 = \frac{1-2\beta}{1-\beta}$$

i.e., $\lim_{f\to 0^+} P\left(D_i[k] \geq 1 \quad \forall k > 0 \mid D_i[0] = 1\right) = \frac{1-2\beta}{1-\beta}$. Now, we can compute $P(V_i^2[r])$ as,

$$\begin{aligned}\lim_{f\to 0^+} P(V_i^2[r]) &= \frac{1-2\beta}{1-\beta}\left(\lim_{f\to 0^+} P(D_i[0]=1)\right) \\ &= \frac{1-2\beta}{1-\beta}(1-\beta) = (1-2\beta)\end{aligned}$$

A.3.3 Final computation of $\phi(f, \beta)$

Having computed the probabilities of $V_i^1[r]$ and $V_i^2[r]$, we can now complete the computation of $\phi(f, \beta)$ as,

$$\begin{aligned}\lim_{f\to 0^+} \phi(f,\beta) &= \lim_{f\to 0^+} P(V_i[r]) = \lim_{f\to 0^+} P(V_i^1[r]) \lim_{f\to 0^+} P(V_i^2[r]) \\ &= \frac{(1-2\beta)}{1-\beta}(1-2\beta) = \frac{(1-2\beta)^2}{1-\beta}\end{aligned}$$

This completes the proof of Lemma A.4. Now, since we want to bound β such that $\phi(f, \beta) > 0.5$ (for a sufficiently small f), we get $8\beta^2 - 7\beta + 1 > 0$. Recall that the Nakamoto consensus properties are not applicable for $\beta > \frac{1}{2}$. Therefore, we end up with the bound $\beta < \frac{7-\sqrt{17}}{16} \approx 0.1798$.

For our (τ, R)-respawning adversaries, recall that another τ fraction of nodes can get killed by the time a transaction is included in all parallel chains, and therefore, we will require $\beta + \tau < \frac{7-\sqrt{17}}{16}$. We also node that this bound may not be tight and that a larger adversarial threshold might still be acceptable.

A.4 Achieving cross-chain-quality

Equipped with the computation of $\phi(f, \beta) = P(V_i[r])$ from the previous section, we can proceed to calculate a bound on the number of chains g. For this, we first declare an event that needs to happen with negligible probability.

An unlikely event. Consider the following event T for any $\epsilon > 0$.

$$T := \left\{ \exists r \in \{1, \cdots, |\mathsf{view}|\} : \frac{1}{g}\sum_{i=1}^{g} \mathbf{1}_{V_i[r]} < \frac{1}{2} - \epsilon \right\}$$

The event T implies that if any transaction is revealed to the public in round r, then in at most $\frac{1}{2} - \epsilon$ fraction of the chains, an honest block will contain the transaction. In the following lemma, we show that if $\phi(f, \beta) > \frac{1}{2}$, then for sufficiently large g, $P(T)$ will be negligible in κ.

Lemma A.6. *Suppose β and a sufficiently small constant mining rate f are appropriately chosen such that $\phi(f,\beta) > \frac{1}{2}$. Then, $P(T) \leq |\mathsf{view}|\, e^{-\Omega(g)}$.*

Proof. Using the union bound,

$$\begin{aligned} P(T) &= P\left(\exists r \in \{1,\cdots,|\mathsf{view}|\} : \frac{1}{g}\sum_{i=1}^{g} \mathbf{1}_{V_i[r]} < \frac{1}{2} - \epsilon\right) \\ &\leq \sum_{1\leq r\leq|\mathsf{view}|} P\left(\frac{1}{g}\sum_{i=1}^{g} \mathbf{1}_{V_i[r]} < \frac{1}{2} - \epsilon\right) \\ &= \sum_{1\leq r\leq|\mathsf{view}|} P\left(\sum_{i=1}^{g} \mathbf{1}_{(V_i[r])^C} > \left(\frac{1}{2} + \epsilon\right) g\right) \end{aligned} \tag{2}$$

where $(V_i[r])^C$ is the complement of $V_i[r]$. Note that $P(\mathbf{1}_{(V_i[r]^C)}) = 1-\phi(f,\beta)$. Since the mining for all chains is independent, we can consider $\mathbf{1}_{(V_1[r])^C}, \cdots, \mathbf{1}_{(V_g[r])^C}$ to be independent of each other. As $1-\phi(f,\beta) < \frac{1}{2}$, therefore, for some $\delta > 0$, we can write

$$\begin{aligned} (1+\delta)g(1-\phi(f,\beta)) &= \left(\frac{1}{2} + \epsilon\right) g \\ \delta &= \frac{\left(\frac{1}{2} + \epsilon\right)}{1-\phi(f,\beta)} - 1 \end{aligned}$$

Then, using the Chernoff bound on equation 2, we have

$$\begin{aligned} P(T) &\leq \sum_{1\leq r\leq|\mathsf{view}|} e^{-\frac{\delta^2 g[1-\phi(f,\beta)]}{3}} \\ &\leq \sum_{1\leq r\leq|\mathsf{view}|} e^{-\Omega(g)} \\ &= |\mathsf{view}|\, e^{-\Omega(g)} \end{aligned}$$

□

Now, note that since $|\mathsf{view}|$ is polynomial in κ, setting $g = \Theta(\kappa)$ results in $P(T)$ being negligible in κ.