Securing Parallel-chain Protocols under Variable Mining Power

Xuechao Wang
xuechao2@illinois.edu
University
of Illinois Urbana-Champaign

Viswa Virinchi Muppirala
virinchi@uw.edu
University of Washington at Seattle

Lei Yang
leiy@csail.mit.edu
MIT CSAIL

Sreeram Kannan
ksreeram@uw.edu
University of Washington at Seattle

Pramod Viswanath
pramodv@illinois.edu
University
of Illinois Urbana-Champaign

ABSTRACT

Several emerging proof-of-work (PoW) blockchain protocols rely on a "parallel-chain" architecture for scaling, where instead of a single chain, multiple chains are run in parallel and aggregated. A key requirement of practical PoW blockchains is to adapt to mining power variations over time (Bitcoin's total mining power has increased by a 10^{14} factor over the decade). In this paper, we consider the design of provably secure parallel-chain protocols which can adapt to such mining power variations.

The Bitcoin difficulty adjustment rule adjusts the difficulty target of block mining periodically to get a constant mean inter-block time. While superficially simple, the rule has proved itself to be sophisticated and successfully secure, both in practice and in theory [11, 13]. We show that natural adaptations of the Bitcoin adjustment rule to the parallel-chain case open the door to subtle, but catastrophic safety and liveness breaches. We uncover a meta-design principle that allow us to design variable mining difficulty protocols for three popular PoW blockchain proposals (Prism [3], OHIE [26], Fruitchains [21]) inside a common rubric.

The principle has three components: (M1) a pivot chain, based on which blocks in all chains choose difficulty, (M2) a monotonicity condition for referencing pivot chain blocks and (M3) translating additional protocol aspects from using levels (depth) to using "difficulty levels". We show that protocols employing a subset of these principles may have catastrophic failures. The security of the designs is also proved using a common rubric – the key technical challenge involves analyzing the interaction between the pivot chain and the other chains, as well as bounding the sudden changes in difficulty target experienced in non-pivot chains. We empirically investigate the responsivity of the new mining difficulty rule via simulations based on historical Bitcoin data, and find that the protocol very effectively controls the forking rate across all the chains.

1 INTRODUCTION

Scaling problem. Built on the pioneering work of Nakamoto, Bitcoin [18] is a permissionless blockchain operating on proof-of-work based on the Nakamoto protocol. The Nakamoto longest-chain protocol was proven to be secure as long as the adversary controlled less than 50% of the mining power in the breakthrough work [11]. Recent works [9, 15, 24] have tried to improve the scalability of Bitcoin [5, 7], in particular the throughput and latency, by redesigning the core consensus protocol. A variety of approaches have been proposed, for example hybrid consensus algorithms [14, 17, 22, 23] try to convert the permissionless problem into a permissioned consensus problem by subselecting a set of miners from a previous epoch. While such approaches achieve scalability, they are not natively proof-of-work (PoW) and hence do not retain the dynamic availability, unpredictability and security against adaptive adversaries that the Nakamoto longest chain protocol enjoys.

Parallel-chain protocols. An emerging set of proof-of-work protocols maintain the native PoW property of Bitcoin and achieve provable scaling by using *many parallel* chains. The chains run in parallel and use an appropriate aggregation rule to construct an ordered ledger of transactions out of the various parallel chains. We will highlight three examples of parallel-chain protocols (PCP): (1) Prism [3], which achieves high-throughput and low-latency using a proposer chain and many voter chains, (2) OHIE [26], which achieves high-throughput using parallel chains and (3) FruitChains [21], which achieves fairness using two distinct types of blocks (blocks and fruits) mined in parallel. There are other approaches such as ledger combiners [10], which achieve some of the same goals using different architectures.

Common structure of PCP. In all of these parallel-chain protocols (PCP), there are multiple types of blocks (for example, in OHIE, each type may correspond to a different chain) and we determine the final type only after mining the block - we will term this process as *hash sortition*. The idea of sortition was first formalized in [11] called 2-for-1 PoW. All three PCPs adopt this technique to achieve parallel mining. A miner creates a single commitment (for example, a Merkle root) to the potential version of the different block types and performs a mining operation. Depending on the region the hash falls, the block is considered mined of a certain type. Different protocols utilize different types of aggregation rules and semantics in order to consider the final ledger out of these parallel chains.

Variable mining power problem. A key requirement of deployed PoW blockchains is to adapt to the immense variation in mining power. For example, the mining power of Bitcoin increased exponentially by an astonishing factor of 10^{14} during its decade of deployment. If Bitcoin had continued to use the same difficulty for the hash puzzle, then the inter-block time would have fallen from the original 10 minutes to 6 **picoseconds**. Such a drop would have caused an intolerable forking rate and seriously undermined the security of Bitcoin, lowering the tolerable adversarial mining power from nearly

Correspondence can be sent to ksreeramw@uw.edu.

50% to 10^{-11}. However, this is prevented by adjusting the difficulty threshold of Bitcoin using a difficulty adjustment algorithm.

Bitcoin difficulty adjustment algorithm. There are three core ideas to the Bitcoin difficulty adjustment algorithm: (a) vary the difficulty target of block mining based on the median inter-block time from the previous epoch (of 2016 blocks), (b) use the *heaviest* chain (calculated by the sum of the block difficulties) instead of the longest chain to determine the ledger, and (c) allow the difficulty to be adjusted only mildly every epoch (by an upper bound of a factor of 4). While this appears to be a simple and intuitive algorithm, minor seemingly-innocuous variants turn out to be dangerously insecure.

Difficulty adjustment terminology. Throughout the paper, we call the hash puzzle threshold in PoW mining the *target* of a block. The *block difficulty* of each block is measured in terms of how many times the block is harder to obtain than using the initial target of the system that is embedded in the genesis block. However, for simplicity, we will adapt the notation of block difficulty to be the inverse of the target of the block. The *chain difficulty* of a chain is the sum of block difficulties of all blocks that comprise the chain, then each block in the chain *covers* an interval of chain difficulty. The chain with the largest chain difficulty is said to be the *heaviest* chain. We also refer the chain difficulty of a block as the chain difficulty of the chain ending at this block. This notation is summarised in the following table.

Target	Threshold of the hash puzzle in PoW mining
Block difficulty	Inverse of the target of a block
Chain difficulty	Sum of block difficulties of all blocks in the chain

Difficulty adjustment requires nuanced design. Consider a simpler algorithm using only (b), i.e., simply let the nodes choose their own difficulty and then use (b) the heaviest chain rule. At a first glance, this rule appears kosher - the heaviest chain rule seems to afford no advantage to any node to manipulate their difficulty. However, this lack of advantage only holds in expectation, and the variance created by extremely difficult adversarial blocks can thwart a confirmation rule that confirms deeply-embedded blocks, no matter how deep, with non-negligible probability proportional to the attacker's mining power (refer to Appendix A for a detailed discussion). Now consider a more detailed rule involving only (a) and (b). It turns out that there is a difficulty raising attack (refer to Appendix A for a detailed discussion), where the adversary creates an epoch filled with timestamps extremely close-together, so that the difficulty adjustment rule from (a) will set the difficulty extremely high for the next epoch, at which point, the adversary can utilize the high variance of the mining similar to the aforementioned attack. This more complex attack is only thwarted using the full protocol that employs (a), (b) and (c) together. The full proof of the Nakamoto heaviest chain protocol was obtained in a breakthrough work [12].

Difficulty adjustment in PCP. When there are multiple parallel-chains, one natural idea is to apply Bitcoin's difficulty adjustment algorithm to each of the chains independently. However, this idea does not integrate well with hash sortition since the range of a particular chain will depend on the state of other chains. Instead, since the mining power variation is the same across all chains, a natural approach is to use *the same difficulty threshold* across all chains, which is then modulated based on past evidence. How should this common difficulty threshold be chosen? One approach is to utilize inter-block arrival times across all the chains to get better statistical averaging and respond faster to mining power variation. However, it requires some sort of synchronization across the chains and breaks the independence assumption.

General methodology. We propose a general methodology by which to adapt parallel-chain architectures to the variable mining rate problem. Our general methodology is comprised of three parts, as detailed below.

- **M1: Pivot-chain.** Use a single chain as the pivot chain for difficulty adjustment. Blocks mined in any other chain need to refer to a block in the pivot chain and use the target inferred therefrom.
- **M2: Monotonicity.** In a non-pivot chain, blocks can only refer to pivot-chain blocks of non-decreasing chain difficulty.
- **M3: Translation.** Wherever the protocol uses the concept of a block's level, it is updated to refer to the block's chain difficulty instead.

Using **M1** pivot-chain for difficulty adjustment ensures that we can continue to use the hash-sortition method. The **M2** monotonicity rule ensures that blocks in non-pivot chain do not refer to stale/old pivot blocks with target which is very different from expected in the present round. Finally, the **M3** translation rule ensures that other aspects of the protocol, such as the confirmation rule are adapted correctly to deal with the variable difficulty regime correctly. We show in Section 3 why each of the three aspects of our methodology is critical in designing variable difficulty for Prism by showing attacks for subsets of **M1,M2,** and **M3**.

On the positive side, we show a concrete adaptation of our general methodology to various schemes, in particular to Prism in Section 3, to OHIE in Section 4 and to FruitChains in Section 5.

Security proofs. The problem of analyzing the difficulty adjustment mechanism in Bitcoin was first addressed in [12] in the lock-step synchronous communication model. It introduces a setting where the number of participating parties' rate of change in a sequence of rounds is bounded but follows a predetermined schedule. Later two concurrent works [6, 13] analyzed the problem in a bounded-delay network with an adaptive (as opposed to predetermined) dynamic participation, with different proof techniques. Following the two later papers, we adopts the more general network and adversary models: we assume a Δ-synchronous communication model, where every message that is received by a honest node is received by all other honest nodes within Δ rounds; we allow the adversary to control the mining rate even based on the stochastic realization of the blockchain, as long as the mining rate does not change too much in a certain period of time. We assume that the adversarial nodes are Byzantine and they do not act rationally. Under this general model, we establish that our proposed modification to Prism, OHIE and FruitChains satisfy the dual security properties of safety and liveness. The proofs require a new understanding of how difficulty evolution in a non-pivot chain progresses based on the difficulty in the pivot chain - this statistical coupling presents a significant barrier to surmount in our analysis, and differs from previous work in this area. We show these results in Section 6.

Systems implementation. Our variable difficulty scheme does not add significant computation and communication overhead on existing parallel-chain protocols, making our protocol an easy upgrade. We conduct extensive simulation studies to examine how our systems respond to varying mining power. Results show that our scheme is able to closely match the system mining power and the mining difficulty for each individual chain, thus keeping the chain forking rate stable. We examine adversarial behavior and how it can influence the difficulties of various chains, and confirm that our scheme is secure against significant adversarial presence. The simulations are based on historical Bitcoin mining power data and parameters collected from real-world experiments of the Prism [25] parallel-chain protocol, making the insights meaningful for real-world systems.

Other related works. A recently proposed blockchain protocol Taiji [16] combines Prism with a BFT protocol to construct a dynamically available PoW protocol which has almost deterministic confirmation with low latency. Since Taiji inherits the parallel-chain structure from Prism, our meta-principles will also apply. The vulnerability of selfish mining has recently been discussed on several existing blockchain projects with variable difficulty in [19]. Our proposed variable difficulty FruitChains protocol guarantees fairness of mining, thus disincentivizes selfish mining.

2 MODEL

Synchronous network. We describe our protocols in the now-standard Δ-synchronous network model considered in [2, 13, 20] for the analysis of proposed variable difficulty protocols, where there is an upper bound Δ in the delay (measured in number of rounds) that the adversary may inflict to the delivery of any message. Observe that notion of "rounds" still exist in the model (since we consider discretized time), but now these are not synchronization rounds within which all messages are supposed to be delivered to honest parties.

Similar to [13, 20], the protocol execution proceeds in "round" with inputs provided by an environment program denoted by $\mathcal{Z}(1^{\kappa})$ to parties that execute the protocol Π, where κ is a security parameter. The adversary $\mathcal{A}$ is adaptive, and allowed to take control of parties on the fly, as well as "rushing", meaning that in any given round the adversary gets to observe honest parties' actions before deciding how to react. The network is modeled as a diffusion functionality similar to those in [13, 20]: it allows order of messages to be controlled by $\mathcal{A}$, i.e., $\mathcal{A}$ can inject messages for selective delivery but cannot change the contents of the honest parties' messages nor prevent them from being delivered beyond Δ rounds of delay — a functionality parameter.

Random oracle. We abstract the hash function as a random oracle functionality. It accepts queries of the form $(\texttt{compute}, x)$ and $(\texttt{verify}, x, y)$. For the first type of query, assuming x was never queried before, a value y is sampled from $\{0,1\}^{\kappa}$ and it is entered to a table T_H. If x was queried before, the pair (x,y) is recovered from T_H. In both cases, the value y is provided as an answer to the query. For the second type of query, a lookup operation is performed on the table. Honest parties are allowed to ask one query per round of the type `compute` and unlimited queries of the type `verify`. The adversary $\mathcal{A}$ is given a bounded number of `compute` queries per round and also unlimited number of `verify` queries. The bound for the adversary is determined as follows. Whenever a corrupted party is activated the bound is increased by 1; whenever a query is asked the bound is decreased by 1 (it does not matter which specific corrupted party makes the query).

Adversarial control of variable mining power. We assume no rational node in the adversarial model. The adversary can decide on the spot how many honest parties are activated adaptively. In a round r, the number of honest parties that are active in the protocol is denoted by n_r and the number of corrupted parties controlled by $\mathcal{A}$ in round r is denoted by t_r. Note that n_r can only be determined by examining the view of all honest parties and is not a quantity that is accessible to any of the honest parties individually. We make the "honest majority" assumption, i.e., $t_r < (1-\delta)n_r$ for all r, where the positive constant $\delta < 1$ is the advantage of honest parties. Further, we will restrict the environment to fluctuate the number of parties in a certain limited fashion. Suppose $\mathcal{Z},\mathcal{A}$ with fixed coins produces a sequence of parties n_r, where r ranges over all rounds of the entire execution, we define the following notation.

Definition 2.1. *Let $r_{\max} \in \mathbb{N}$ is the total number of rounds in the execution. For $\gamma \in \mathbb{R}^+$, we call $(n_r), r \in [0, r_{\max}]$, as (γ,s)-respecting if for any set $S \subseteq [0, r_{\max}]$ of at most s consecutive rounds,*

$$\max_{r \in S} n_r \le \gamma \min_{r \in S} n_r.$$

We say that $\mathcal{Z}$ is (γ,s)-respecting if for all $\mathcal{A}$ and coins for $\mathcal{Z}$ and $\mathcal{A}$ the sequence of honest parties n_r is (γ,s)-respecting.

3 PRISM

3.1 Fixed Difficulty Algorithm

Each block in the longest chain of the Bitcoin protocol performs dual roles: Proposing and Voting. A proposed block gets confirmed with high reliability only after the block, and several more blocks extending it make it in the longest chain. The latency of the protocol is the number of blocks for which one needs to wait (this number depends on the reliability). To guarantee security, the mining rate remains low [11], which leads to low throughput and high latency. Prism [3] is a Proof-of-Work protocol that decouples block proposals and voting to scale throughput and latency. We will briefly explain the Prism as it was originally described in the fixed mining rate, i.e., fixed difficulty regime. As show in Figure 1, Prism runs multiple $m+1$ separate parallel "blocktrees" where one of the trees, called the proposer tree, consists of blocks from which the final transaction ledger is constructed. We define a block's level in the proposer tree as the block's depth from the genesis. The final transaction ledger will comprise of one proposer block at each level chosen by the longest chain in each of the m voter blocktrees; they are referred to as voter chains. A voter block in any voter blocktree can vote for one or more proposer blocks at different levels by including a pointer to the corresponding proposer blocks in its payload. A voter block can also consist of a null vote if the chain it entered already voted on the latest level. At a given level of the proposer tree, a voter chain can vote for exactly one proposer block. The net vote for a proposer block can be counted by aggregating which of the m voter chains voted for that block. The block with the most votes at any particular level is termed as the leader block for that level, and the ledger is constructed by concatenating the leader blocks at various levels.

Mining and sortition. In order to ensure that the adversary cannot focus the mining power onto a single chain, a "sortition" mechanism is used. A miner creates a "super-block" containing information

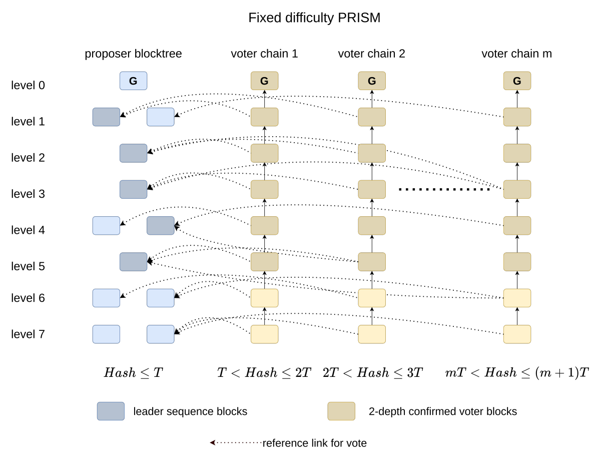


Figure 1: Fixed difficulty Prism. **Snapshot of a miner's view of** Prism **block-trees. The confirmed blocks are darker in color and the votes are shown using dotted arrows.**

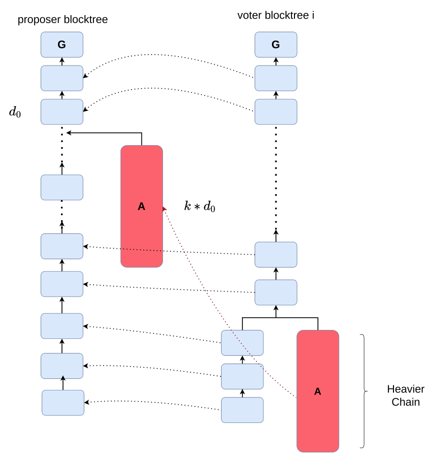


Figure 2: Attack on safety when we enforce M1 but not M2. At the current time, the adversary will choose a very difficult proposer block in a less heavier chain as its proposer-parent in the voter chain, hurting the ledger's security. The dotted arrows represent the relation between the voter blocks and their proposer parents.

about its parent block in each of the $m+1$ trees. Each tree has a target of T, and for $1 \leq i \leq m$ if a node creates a block of hash value in between iT and $(i+1)T$, it will be able to mine this block in the voter block-tree i. If it creates a block of hash value less than T, it will be able to mine this block in the proposer block-tree as show in Figure 1.

The structure of this voting scheme in Prism enables low confirmation latency. The high-level idea is that votes accrue only sequentially in Bitcoin, whereas in Prism, votes accrue parallelly. Thus for a given amount of security, confirmation only needs to wait for a much shorter amount of time (since voting blocks are created in parallel), thus reducing the amount of latency.

3.2 Natural Approaches Are Insecure

Different difficulty adjustment for different chains (no M1). To add support for variable mining power to Prism, a natural first approach is to replace the longest chain rule [11] by the heaviest chain rule [13] in all the parallel chains, and adjust the mining difficulty in each chain separately. However, miners in Prism use cryptographic sortition to mine blocks on all chains at the same time, and having different thresholds for different chains depending on the state will require complex coupling across chains. Furthermore, since the mining power variation is the same across different chains, it is efficient to have a single difficulty threshold across the entire system.

As we explained in the introduction, our general methodology for converting a fixed-difficulty protocol into a variable-difficulty protocol comprises of three attributes **M1**: Pivot-chain, **M2**: Monotonicity and **M3**: Translation. We will now explore the subtleties inherent in this process and show why a subset of these attributes is insufficient for Prism.

M1 without M2 $\Rightarrow$ Safety failure. To make the cryptographic sortition technique applicable, a straightforward approach is to use the proposer chain as a pivot chain for difficulty adjustment (**M1**): the difficulty of the proposer blocks is adjusted according to the Bitcoin rule [13], and the difficulty of voter blocks tracks that of the proposer chain by reference links. However, if we allow the miners to use the difficulty of any proposer block for the voter blocks, then **safety failures** may occur on the voter chains. We demonstrate an example safety failure here. Let the honest parties maintain the difficulty d_0 throughout the execution, and the adversary mines private proposer blocks with timestamps in rapid succession to increase the difficulty to $d_0 * k$, where k is a desired security parameter on the voter chains (i.e., we hope that a k-deep voter block will be stable forever). Even if the adversary cannot keep up the chain difficulty of its private proposer chain with the heaviest public chain, at the current time on the voter chain, the adversary will refer to this very difficult block on the proposer tree to create a very difficult block on the voter tree. If the adversary is lucky and mines one voter block with difficulty $d_0 * k$, the probability of which is a constant rather than exponentially decaying in k (via the same anti-concentration argument in Appendix A), then it can overtake the heaviest voter chain and reverse a k-deep voter block. This attack is described in Figure 2. To address this issue, we require that on each voter chain the referred proposer blocks should have non-decreasing chain difficulty (**M2**), so that the adversary can no longer adopt an old mining difficulty from the proposer chain. With **M2**, although the adversary may not refer to the tip of the proposer chain, both our analysis in Section 6.4 and the simulation in Section 7.3 show that the security of the voter chain can still be guaranteed.

Voting rule: No M3 $\Rightarrow$ Liveness Failure. In fixed difficulty Prism, a voter block votes on all levels in the proposer tree that are unvoted by the voter block's ancestors. In the variable difficulty algorithm, while the notion of "level" on the proposer chain is well-defined an adversary can always mine a very long but easy proposer chain. As a result, if we still order proposer blocks by level, the leader sequence will be full of adversarial blocks, which may cause liveness failure as described in Figure 3. A natural generalization would be that voter blocks vote for each difficulty value rather than level and the leader sequence is also decided for each difficulty value (**M3**). See the complete algorithm in the next subsection.

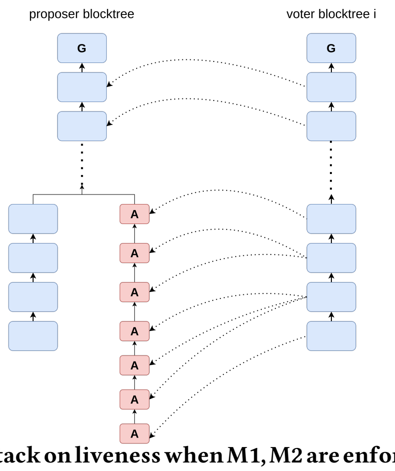


Figure 3: Attack on liveness when M1, M2 are enforced but not M3. The adversary lowers the block difficulty and advances in level on the proposer tree. The voter blocks will not be able to vote for any honest blocks, hurting the ledger's liveness.

3.3 Variable Difficulty Algorithm

We now describe the full Prism protocol for the variable difficulty setting constructed using our general methodology. We refer the reader to Appendix F for a pseudocode of the algorithm. There are two types of blocks in Prism blockchain: proposer blocks and voter blocks. Proposer blocks contain transactions that are proposed to be included in the ledger, and constitutes the skeleton of Prism blockchain. Voter blocks are mined on m separate voter blocktrees, each with its own genesis block. We say a voter block votes on a proposer block B if it includes a pointer to B in its payload.

Block proposal rule. The proposer chain follows the heaviest chain rule, and the difficulty adjustment uses the target calculation function defined in [13] with parameter Φ and τ, where Φ is the length of an epoch in number of blocks and $\tau \geq 1$ is the dampening filter (line 14 in Algorithm 2). All m voter chains also follow the heaviest chain rule, but the difficulty adjustment on voter chains is more tricky and we will discuss it soon when introducing *sortition.*

Whereas Bitcoin miners mine on a single blocktree, Prism miners simultaneously mine one proposer block and m voter blocks via cryptographic sortition. More precisely, while mining, each miner selects $m+1$ parent blocks, which are the tips of the heaviest chains on the proposer tree and the m voter trees. We call these tips *proposer parent* and *voter parents* separately. And the miner maintains outstanding content for each of the $m+1$ possible mined blocks: For the proposer block, the content is a list of transactions; For the voter block on the i-th voter tree, the content is a list of hashes of proposer blocks at each difficulty in the proposer blocktree that has not yet received a vote in the heaviest chain of the i-th voter tree. More precisely, on the i-th voter tree, if the last proposer block voted by the heaviest chain covers the difficulty interval $(a_0,b_0]$ and the proposer parent covers $(a^*,b^*]$, then a valid voter block on the i-th voter tree must satisfy the following conditions (see Algorithm 4).

- If $b^* = b_0$, then it should contain no vote.
- If $b^* > b_0$, then it should vote for an arbitrary number of proposer blocks $B_1,B_2,\cdots,B_n$, each covering $(a_1,b_1]$, $(a_2,b_2]$, $\cdots$, $(a_n,b_n]$, such that $a_i < b_{i-1} < b_i$ for all $1 \leq i \leq n$ and $b_n = b^*$.

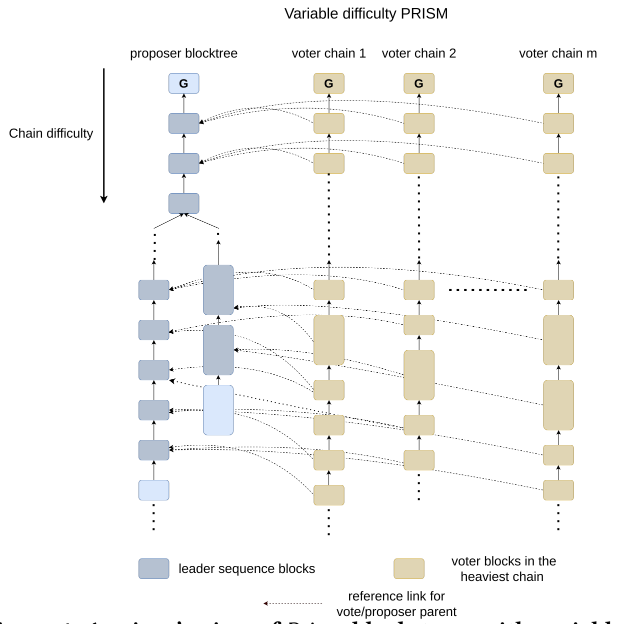


Figure 4: A miner's view of Prism **block-trees with variable chain difficulty. The confirmed blocks are darker in color. Both the votes and proposer-parent links are shown using the same dotted arrows.**

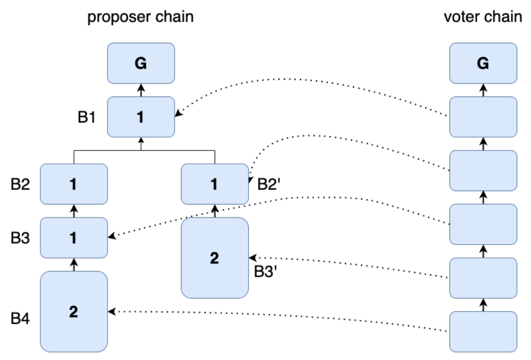
Upon collecting this content, the miner tries to generate a block with target according to the proposer parent via proof-of-work (**M1**). Once a valid nonce is found, the output of the hash is deterministically mapped to either a voter block in one of the m trees or a proposer block (lines 19-25 in Algorithm 2).

While mining, nodes may receive blocks from the network, which are processed in much the same way as Bitcoin. For a received voter block to be valid, the chain difficulty of its proposer parent must be at least that of the proposer parent of its voter parent (**M2**). Upon receiving a valid voter block, the miner updates the heaviest chain if needed, and updates the vote counts accordingly. Upon receiving a valid proposer block B with chain difficulty higher than the previous heaviest chain, the miner makes B the new proposer parent, and updates all m voter trees to vote for chain difficulties until B.

Ledger formation rule. Note that all the voters on one voter chain may cover overlapping intervals. So we first sanitize them into disjoint intervals: For n consecutive valid votes $(a_1,b_1]$, $(a_2,b_2]$, $\cdots$, $(a_n,b_n]$ on a voter chain, we sanitize them into new intervals $(a_1,b_1],(b_1,b_2],\cdots,(b_{n-1},b_n]$. In this way, we make sure that each real-valued difficulty d is voted at most once by each voter chain, hence d can receive at most m votes. Since voter blocks vote for each difficulty value rather than level, the ledger is also generated based on difficulty values (**M3**). Let $v_i(d)$ be the proposer block with interval containing d voted by the heaviest chain on the i-th voter tree. Let $\ell(d)$ be the leader block of difficulty d, which is the plurality of the set $\{v_i(d)\}_{i=1}^{m}$. For each proposer block B_p in the proposer tree, define $g(B_p)$ as

$$g(B_p) = \inf_{d\geq 0} \{d : \ell(d) = B_p\}. \tag{1}$$

Note that if $\{d : \ell(d) = B_p\}$ is empty, then $g(B_p) = \infty$. Finally, by sorting all proposer blocks by $g(\cdot)$, we get the leader sequence of the



d	(0,1]	(1,2]	(2,3]	(3,4]	(4,5]
$\ell(d)$	B1	B2'	B3	B3'	B4

B	B1	B2	B3	B4	B2'	B3'
g(B)	0	∞	2	4	1	3

Leader sequence: B1 B2' B3 B3' B4

Figure 5: An example of the ledger formation rule in Prism**. For simplicity, we only have one voter chain in the example. The number inside each proposer block is the block difficulty. In this example, the heaviest proposer chain has chain difficulty 5. We find the leader block $\ell(d)$ for each difficulty level d in $(0, 5]$ according to the votes (as shown in the first table). Then we find the grade $g(\cdot)$ of each proposer block by Equation (1) as shown in the second table. Finally, the proposer blocks are ordered by their grades.**

proposer blocks. A concrete example of this ledger formation rule is shown in Figure 5.

Operationally, we only need to count votes for intervals in the atomic partition of all intervals covered by the proposer blocks. After finding the leader block for each atomic interval, we can get the leader sequence by sanitizing the repeated proposer blocks.

Main result: persistence and liveness of Prism **(Informal)** We show that Prism generates a transaction ledger that satisfies *persistence* and *liveness* in a variable mining power setting in Theorem 6.17.

4 OHIE

4.1 Fixed Difficulty Algorithm

OHIE [26] composes m parallel instances of Bitcoin longest chains. Each chain has a distinct genesis block, and the chains have ids from 0 to $m-1$. Similar to Prism, OHIE also uses cryptographic sortition to ensure that miners extend the m chains concurrently and they do not know which chain a new block will extend until the PoW puzzle is solved.

Each individual chain in OHIE inherits the proven security properties of longest chain protocol [11], and all blocks on the m chains confirmed by the longest chain confirmation rule (eg. the k-deep rule) are called *partially-confirmed*. However, this does not yet provide a total ordering of all the confirmed blocks across all the m chains in OHIE. The goal of OHIE is to generate a *sequence of confirmed blocks*

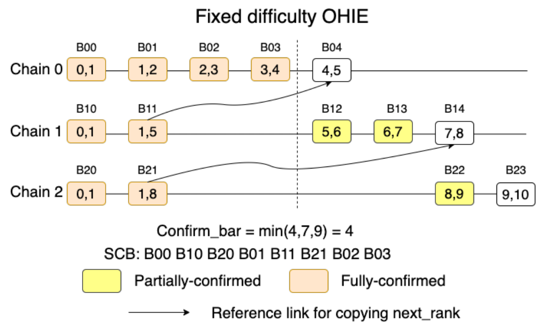


Figure 6: OHIE **with fixed difficulty. Each block has a tuple** (rank,next_rank)**. In this figure, a block that is at least 2-deep in its chain is partially-confirmed. The blocks arrive in this order: B00, B10, B20, B01, B02, B03, B04, B11, B12, B13, B14, B21, B22, B23.**

(SCB) across all m parallel chains. Once a partially-confirmed block is added to SCB, it becomes *fully-confirmed*.

In OHIE, each block has two additional fields used for ordering blocks across chains, denoted as a tuple (rank,next_rank). In SCB, the blocks are ordered by increasing rank values, with tie-breaking based on the *chain ids*. For any new block B that extends from its parent block denoted as parent(B), we directly set B's rank to be the same as parent(B)'s next_rank. A genesis block always has rank of 0 and next_rank of 1. Properly setting the next_rank of a new block B is the key design in OHIE. Let $\mathcal{B}$ be the set of all tips of the m longest chains before B is added to its chain, then the next_rank of B is given by

$$\texttt{next\_rank}(B) = \max\{\texttt{rank}(B)+1, \max_{B' \in \mathcal{B}}\{\texttt{next\_rank}(B')\}\}.$$

If B copies the next_rank of a block B' on a chain with different id, then a reference link to B' (or the hash of B') is added into B. In the example of Figure 6, when B11 is mined, B04 has the highest next_rank, so B11 copies the next_rank of B04 and has a reference link to B04.

OHIE generates a SCB in the following way. Consider any given honest node at any given time and its local view of all the m chains. Let y_i be the next_rank of the last partially-confirmed block on chain i in this view. Let confirm_bar $\leftarrow \min_{i=1}^{k} y_i$. All partially-confirmed blocks whose rank is smaller than confirm_bar are deemed fully-confirmed and included in SCB. Finally, all the fully-confirmed blocks will be ordered by increasing rank values, with tie-breaking favoring smaller chain ids. As an example, in Figure 6, we have $y_0 = 4, y_1 = 7, y_2 = 9$, hence confirm_bar is 4. Therefore, the 8 partially-confirmed blocks whose rank is below 4 become fully-confirmed.

4.2 Variable Difficulty Algorithm

Following the same meta-principle of designing variable difficulty Prism, we can also turn the fixed difficulty OHIE into a variable difficulty algorithm by making the following changes.

- Each individual chain follows the heaviest chain rule instead of the longest chain rule.
- The mining difficulty of chain 0 is adjusted the same way as the Bitcoin rule [13].

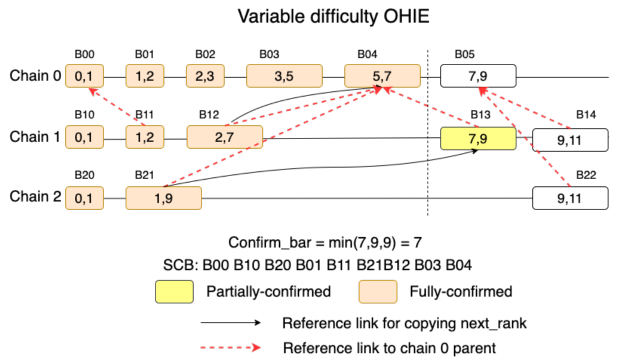


Figure 7: OHIE with variable difficulty. Each block has a tuple (rank,next_rank)**. In this figure, a block that is at least 2-deep in its chain is partially-confirmed. The width of a block represents its mining difficulty. Different from the fixed difficulty algorithm, the mining difficulty is adjusted every 3 blocks on chain 0; Each block B on chain i ($i > 0$) has a chain 0 parent (shown by the red reference link), which decides the mining difficulty of B. The blocks arrive in this order: B00, B10, B20, B11, B01, B02, B03, B04, B12, B13, B21, B05, B14, B22.**

- Following our design principle **M1**, each block B on chains $1, 2, \ldots, (m-1)$ will also have a *chain 0 parent* $\hat{B}$ (assigned before mining). The mining difficulty of B is the same as the difficulty used to mine a child block of $\hat{B}$. To prevent the adversary from adopting an old mining difficulty from chain 0, we require that on each chain the referred chain 0 parent should have non-decreasing chain difficulty (**M2**). As an example in Figure 7, each block on chain 1 and chain 2 refers to (shown in red dashed arrow) a chain 0 parent with non-decreasing chain difficulty, which decides the mining difficulty of the block.
- A straightforward adoption on how to decide the `next_rank` of a block would follow from our design principle **M3**. Let $\mathcal{B}$ be the set of all tips of the m heaviest chains before B is added to its chain, then the `next_rank` of B is given by

$$\texttt{next\_rank}(B) = \max\{\texttt{rank}(B) + \text{diff}(B), \max_{B' \in \mathcal{B}}\{\texttt{next\_rank}(B')\}\}.$$

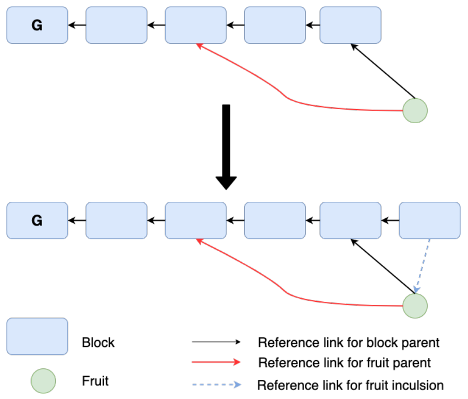
  If B copies the `next_rank` of a block B' on a chain with different id, then a reference link to B' (or the hash of B') is added into B. Note that B' may be different from B's chain 0 parent, eg. B21 in Figure 7. We point out that this design is not necessary for the security analysis, but it is a very natural choice.

Main result: persistence and liveness of OHIE **(Informal)** We show that OHIE generates a transaction ledger that satisfies *persistence* and *liveness* in a variable mining power setting in Theorem D.2.

5 FRUITCHAINS

5.1 Fixed Difficulty Algorithm

The FruitChains protocol was developed in order to solve the selfish mining problem and develop incentives which are approximately a Nash equilibrium. A key underlying step in FruitChains is to ensure that a node that controls a certain fraction of mining power receives reward nearly proportional to its mining power, irrespective of adversarial action. FruitChains runs an instance of Nakamoto consensus but instead of directly putting the transactions inside the blockchain, the transactions are put inside "fruits" and fruits are included by blocks. Mining fruits also requires solving some PoW puzzle. Similar to Prism and OHIE, the FruitChains protocol also uses cryptographic sortition to ensure that miners mine blocks and fruits concurrently and they do not know the type of the blocks until the puzzle is solved. Additionally, a fruit is required to "hang" from a block which is not too far from the block which includes the fruit.



G
G
Block
Fruit
Reference link for block parent
Reference link for fruit parent
Reference link for fruit inculsion

Figure 8: The FruitChains **protocol.**

In FruitChains, each of the fruit will have two parent blocks, we call them fruit parent and block parent: the fruit parent is a recently stabilized/confirmed block that the fruit is hanging from; the block parent should be the tip of the longest chain. A block will also have a fruit parent because the fruit mining and block mining are piggybacked atop each other, but a block actually does not care about this field. See Figure 8 for illustration. We say that a fruit B_f is recent w.r.t. a chain C if the fruit parent of B_f is a block that is at most R deep in C, where R is called the recency parameter. The FruitChains protocol requires that blocks only include recent fruits. Intuitively, the reason why fruits need to be recent is to prevent the "fruit withhold attack": without it, an attacker could withhold fruits, and suddenly release lots of them at the same time, thereby creating a very high fraction of adversarial fruits in some segment of the chain.

We term a blockchain protocol as fair if players controlling a ϕ fraction of the computational resources will reap a ϕ fraction of the rewards. Intuitively, the reason why the FruitChains protocol guarantees fairness is that even if an adversary tries to "erase" some block mined by an honest player (which contains some honest fruits), by the liveness of the longest chain protocol, eventually an honest player will mine a new block including those fruits and the block will be stable – in fact, by setting the recency parameter R reasonably large, we can make sure that any fruit mined by an honest player will be included sufficiently deep in the chain. And further, if rewards and transaction fees are evenly distributed among the fruits in the long segment of the chain, then the FruitChains protocol guarantees fairness.

5.2 Variable Difficulty Algorithm

Following our meta-principles, we can also turn the fixed difficulty FruitChains into a variable difficulty algorithm by making the following changes.

- The underlying blockchain protocol follows the heaviest chain rule instead of the longest chain rule, i.e., the block parent of a block/fruit is the tip of the heaviest chain.
- The mining difficulty is adjusted the same way as the Bitcoin rule [13], and the block/fruit mining will use the same mining difficulty, or the difficulties of fruit and block will remain the same ratio (**M1**).
- A fruit B_f is recent w.r.t. a chain C at round r if the fruit parent of B_f is in C and has timestamp at least $r-R$, where R is called the recency parameter. And again, blocks only include recent fruits, i.e., a block B with timestamp r is valid if for all fruits $B_f \in B$, the fruit parent of B_f has timestamp at least $r-R$.

If rewards and transaction fees are designed to distribute proportional to the fruit difficulty in a sufficiently long segment of the chain, then the variable difficulty FruitChains protocol guarantees fairness under a variable mining power setting. This is where the meta-principle **M3** kicks in. In the fixed difficulty setting, the reward is distributed equally among all fruit miners **equally** in a window of blocks. In the variable difficulty setting, the reward is distributed **proportional to the difficulty of the fruits**. To model this in our calculation of fairnesss, we say that the variable difficulty protocol is fair if the **fraction of difficulty** of fruits of a given miner in a window is approximately proportional to its mining power. Note that monotonicity condition (**M2**) does not apply to variable difficulty FruitChains as there is no chaining structure among the fruits. But the recency condition on the fruits has the same effect and does prevent the adversary from adopting an old mining difficulty for fruits.
Main result: persistence, liveness and fairness of FruitChains **(Informal)** We show that FruitChains generates a transaction ledger that satisfies *persistence*, *liveness* and *fairness* in a variable mining power setting in Appendix E.

6 SECURITY ANALYSIS

6.1 Desired Security Properties

Notation 6.1. *We denote by $C^{\lceil \ell}$ the chain resulting from "pruning" the blocks with timestamps within the last ℓ rounds. If C_1 is a prefix of C_2, we write $C_1 \prec C_2$. The latest block in the chain C is called the head of the chain and is denoted by head(C). We denote by $C_1 \cap C_2$ the common prefix of chains C_1 and C_2. We say that a chain C is held by or belongs to an honest party if it is one of the heaviest chains in its view.*

The following two properties called common prefix and chain quality, are essential in proving the persistence and liveness of the transaction ledger. The common prefix property states that any two honest parties' chains at two rounds have the earlier one subsumed in the later as long as the last a few blocks are removed, while chain quality quantifies the contributions of the honest parties to any sufficiently long segment of the chain.

Definition 6.2 (Common Prefix). *The common prefix property with parameter $\ell_{\text{cp}} \in \mathbb{N}$ states that for any two honest players holding chains C_1, C_2 at rounds r_1, r_2, with $r_1 \le r_2$, it holds that $C_1^{\lceil \ell_{\text{cp}}} \prec C_2$.*

Definition 6.3 (Chain Quality). *The chain quality property is defined for two parameters $\ell_{\text{cq}} \in \mathbb{N}$ and $\mu \in \mathbb{R}$. Let C be a chain held by any honest party at round r and let $S_0 \subseteq [0,r]$ be an interval with at least ℓ_{cq} consecutive rounds. Let $C(S_0)$ be the segment of C containing blocks with timestamps in S_0 and d be the total difficulty of all blocks in $C(S_0)$. The chain quality property states that the honest blocks in $C(S_0)$ have a total difficulty of at least μd.*

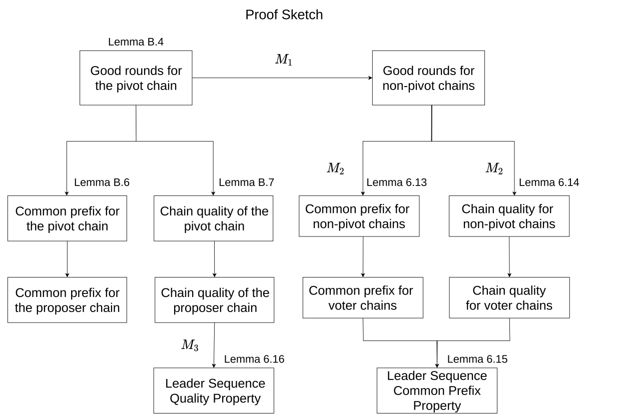

Proof Sketch
Lemma B.4
Good rounds for the pivot chain
M_1
Good rounds for non-pivot chains
Lemma B.6
Lemma B.7
M_2
Lemma 6.13
M_2
Lemma 6.14
Common prefix for the pivot chain
Chain quality of the pivot chain
Common prefix for non-pivot chains
Chain quality for non-pivot chains
Common prefix for the proposer chain
Chain quality of the proposer chain
Common prefix for voter chains
Chain quality for voter chains
M_3
Lemma 6.16
Lemma 6.15
Leader Sequence Quality Property
Leader Sequence Common Prefix Property

Figure 9: Proof sketch for Prism**. M1, M2 and M3 are crucial in proving these properties for the leader sequence.**

In the context of Prism, let $\mathsf{LedSeq}_d(r)$ be the *leader sequence* up to difficulty level d at round r. And the leader sequence at the end of round $r_{\max}$, the end of the protocol execution, is the *final leader sequence*, $\mathsf{LedSeq}_d(r_{\max})$. Then similar to a single chain, we can define the following properties on the leader sequence.

Definition 6.4 (Leader Sequence Common Prefix). *The leader sequence common prefix property with parameter $\ell_{\text{lscp}} \in \mathbb{N}$ states that for a fixed difficulty level d, let R_d be the first round in which a proposer block covering d was received by all honest players, then it holds that*

$$\mathsf{LedSeq}_d(r) = \mathsf{LedSeq}_d(r_{\max}) \quad \forall r \ge R_d + \ell_{\text{lscp}}. \tag{2}$$

Definition 6.5 (Leader Sequence Quality). *The leader sequence property is defined for two parameters $\ell_{\text{lsq}} \in \mathbb{N}$ and $\mu \in \mathbb{R}$. Let C be a proposer chain held by any honest party at round r and let D be the difficulty range covered by all blocks in C with timestamps in the last ℓ_{lsq} rounds. The leader sequence quality property states that leader blocks mined by honest players cover at least μ fraction of D.*

Our goal is to generate a robust transaction ledger that satisfies *persistence* and *liveness* as defined in [11, 26].

Definition 6.6 (from [11, 26]). *A protocol Π maintains a robust public transaction ledger if it organizes the ledger as a blockchain of transactions and it satisfies the following two properties:*

- *(Persistence) Consider the confirmed ledger L_1 on any node u_1 at any round r_1, and the confirmed ledger L_2 on any node u_2 at any round r_2 (here u_1 (r_1) may or may not equal u_2 (r_2)). If $r_1 + \Delta < r_2$, then L_1 is a prefix of L_2.*
- *(Liveness) Parameterized by $u \in \mathbb{R}$, if a transaction* tx *is received by all honest nodes for more than u rounds, then all honest nodes will contain* tx *in the same place in the confirmed ledger.*

6.2 Proof Sketch

Since there is a pivot chain in all three protocols (by **M1**), the first step of our analysis is to prove some desired properties (including chain growth, common prefix, and chain quality) of the pivot chain. As the pivot chain just follows the difficulty adjustment rule as in Bitcoin, we can directly borrow results from a beautiful paper [13]. The key step is to show that by adopting the heaviest pivot chain, honest nodes are always mining with "reasonable" block difficulties (this is formally defined as Good round/chain in Section 6.3). We state all the useful lemmas and summarize the proof from [13] in Appendix B.

The key technical challenge involves analyzing the properties of the non-pivot chains. Unlike in a pivot chain where all blocks in an epoch will have the same block difficulty, the block difficulties may experience sudden changes in non-pivot chains. This presents a significant barrier to surmount in our analysis, and differs from previous work in this area. Recall that **M1** ensures that an honest party chooses the target of the next block in a non-pivot chain from the tip of the heaviest pivot chain in its view. Hence, the targets used by an honest party for the non-pivot chains are also reasonable. Then how about the non-pivot-chain blocks mined by the adversary? As discussed in Section 3, allowing the miners to choose arbitrary mining difficulty in a non-pivot chain is risky. So we use the monotonicity condition **M2** to ensure that non-pivot-chain blocks also have "reasonable" block difficulties even if the adversary mines them.

Then we prove that any two heaviest non-pivot chains cannot diverge for too long to prove the common prefix property. We do this by considering two non-pivot chains C_1 and C_2 (in one of the non-pivot block tree) that diverge for too long and consider the last common honest block B of C_1 and C_2. **M2** ensures that the blocks arriving after B should refer to a pivot-chain block with monotonically non-decreasing chain difficulty than the one referred by B. We also argue that the chain difficulty intervals covered by uniquely successful honest blocks (defined as honest blocks that are mined more than Δ rounds apart) in chains C_1 and C_2 do not overlap similar to the analysis for the common prefix in [13]. To make C_1 and C_2 diverge, the adversary has to accumulate an enormous total difficulty compared to uniquely successful honest blocks.

When the number of adversarial queries is high in the chains C_1 and C_2 after the block B, we bound the difficulty accumulated by the adversary via concentration. When it is low, the variance is high; we prove this by dividing the problem into 5 cases. Since the adversary cannot contribute an enormous total difficulty compared to uniquely successful honest blocks in one of the heaviest chains, the chain quality property also holds. The full proof can be found in Section 6.4.

The last step of our proof is using the desired proprieties on each individual chain to show the security of the full parallel-chain protocol. Since each parallel-chain protocol has its own way of forming the transaction ledger, the proof also has to differ. By properly turning the concept of block level to block's chain difficulty (**M3**), we make sure that our proof works out for all three protocols. We complete the proof of persistence and liveness for Prism in Section 6.5 (and the flowchart of the proof sketch can be found in Figure 9), while the proof for OHIE can be found in Appendix D. In addition, we define and prove block reward fairness of FruitChains under a variable mining setting in Appendix E.

$m\in\mathbb{N}$	number of voter/parallel chains in Prism/OHIE
n_r	number of honest parties mining in round r
t_r	number of corrupted parties mining in round r
δ	advantage of honest parties ($t_r<(1-\delta)n_r$ for all r)
Δ	network delay in rounds
κ	security parameter; length of the hash function output
$\Phi\in\mathbb{N}$	the length of an epoch in number of blocks
$\tau\geq 1$	the dampening filter (Definition 6.7)
(γ,s)	restrictions on the fluctuation of the number of parties across rounds (Definition 2.1)
f	expected mining rate in number of blocks per round
ε	quality of concentration of random variables
λ	related to the properties of the protocol
ℓ	minimum number of rounds for concentration bounds
$r_{\max}$	total number of rounds in the execution

Table 1: The parameters used in our analysis.

6.3 Definitions

Let T,Λ,Φ and n denote the target of a block, duration of an epoch, epoch length and number of honest parties respectfully. Throughout the analysis, the block difficulty of a block with target T is set to be $1/T$. The chain difficulty of a chain is equal to the sum of all block difficulties that comprise the chain. The following is the target recalculation function for the pivot chain which is the same function used in Bitcoin.

Definition 6.7 (from [11]). *Consider a pivot chain of v blocks with timestamps $(r_1...r_v)$. For fixed constants κ,τ,Φ,n_0 the initial number of participants, T_0 the initial target, the target calculation function $\mathcal{T}:\mathbb{Z}^*\rightarrow\mathbb{R}$ is defined as*

$$\mathcal{T}(\emptyset)=T_0,$$
$$\mathcal{T}(r_1...r_v)=\begin{cases}\frac{1}{\tau}T & \text{if } \frac{n_0}{n(T,\Lambda)}T_0<\frac{1}{\tau}T\\ \tau T & \text{if } \frac{n_0}{n(T,\Lambda)}T_0>\tau T\\ \frac{n_0}{n(T,\Lambda)}T_0 & \text{otherwise}\end{cases}$$

where $n(T,\Lambda)=2^{\kappa}\Phi/T\Lambda$, with $\Lambda=r_{\Phi'}-r_{\Phi'-\Phi}$, $T=\mathcal{T}(r_1...r_{\Phi'-1})$, and $\Phi'=\Phi\lfloor v/\Phi\rfloor$.

We now define a notion of "good" properties such as good round and good chain. These properties will bound the targets used by the honest parties, which will help us prove chain quality and common prefix.

Definition 6.8 (Good round, from [13]). *Let T_r^{min} and T_r^{max} denote the minimum and the maximum targets the n_r honest parties are querying the oracle for in round r. Round r is good iff $f/2\gamma^2\leq pn_rT_r^{min}$ and $pn_rT_r^{max}\leq(1+\delta)\gamma^2 f$.*

Definition 6.9 (Good chain, from [13]). *Round r is a target-recalculation point of a pivot chain C, if C has a block with timestamp r and height a multiple of Φ. A target-recalculation point r is good if the target T of the next block satisfies $f/2\gamma\leq pn_rT\leq(1+\delta)\gamma f$. A pivot chain C is good if all its target-recalculation points are good.*

We will use the superscript P to denote the variables, blocks, chains and sets corresponding to the pivot chain/tree and i to denote the ones of the i^{th} non-pivot chain/tree.

At any round r of an execution, the adversary may keep chains in private that have the potential to be adopted by an honest party (because the private chains are heavier than the heaviest chain adopted by the honest party). So, we expand our chains of interest beyond the chains that belong to an honest party. For every non-pivot tree and the pivot tree, we define a set of valid chains $\mathcal{S}_r^P$ and $\mathcal{S}_r^i$ [13] that include the chains that belong to or have the potential to be adopted by an honest party.

We will be dealing with random variables to quantify the difficulty accumulated by the honest parties and the adversary in our analysis. At round r, define the real random variable D_r^P equal to the sum of the difficulties of all pivot-chain blocks computed by honest parties. Also, define Y_r^P to equal the maximum difficulty among all pivot-chain blocks computed by honest parties, and Q_r^P to equal Y_r^P when $D_u^P = 0$ for all $r < u < r + \Delta$ and 0 otherwise. We call an honest block uniquely successful if it is mined at round r such that $Q_r > 0$. Similarly define D_r^i, Y_r^i and Q_r^i for the i-th non-pivot chain ($1 \leq i \leq m$ in Prism and $1 \leq i \leq m-1$ in OHIE). For a set of rounds S, we define $D^P(S) = \sum_{r\in S} D_r^P, Q^P(S) = \sum_{r\in S} Q_r^P$ and $D^i(S) = \sum_{r\in S} D_r^i, Q^i(S) = \sum_{r\in S} Q_r^i$ for all i.

Regarding the adversary, for a set of J adversarial queries to the oracle, let $T(J)$ be target associated with the first query in J. Define the real random variable $A^P(J)$, as the sum of difficulties of all the adversarial blocks created during queries in J with difficulty less than $\tau/T(J)$. For all i, define $A^i(J)$ as the sum of difficulties of all the adversarial blocks created during queries in J with difficulty less than $b^i(J) = max_{j\in J} sup\{A_j^i - A_{j-1}^i | \mathcal{E}_{j-1} = E_{j-1}\}$, a function associated with the set of queries J (defined according to Theorem 8.1 in [8]). A_j^P is the difficulty of the pivot-chain block with difficulty at most $\tau/T(J)$ obtained at the j^{th} query of J. A_j^i is the difficulty of the block obtained at j^{th} query of J for non-pivot chain i.

Let $\mathcal{E}$ denote the entire execution and let $\mathcal{E}_r$ be the execution just before round $r+1$. To obtain meaningful concentration of our random variables, we should be considering a sufficiently long sequence of at least

$$\ell \triangleq \frac{4(1+3\varepsilon)}{\varepsilon^2 f[1-(1+\delta)\gamma^2 f]^{\Delta+1}} \max\{\Delta,\tau\}\gamma^3\lambda \tag{3}$$

consecutive rounds.

We require Φ the duration of an epoch to be large enough in order to obtain meaningful security bounds:

$$\Phi \geq 4(1+\delta)\gamma^2 f(\ell+3\Delta)\varepsilon. \tag{4}$$

In order for the proofs for the security analysis to work, the parameters of the protocol should satisfy the following conditions:

$$[1-(1+\delta)\gamma^2 f]^{\Delta} \geq 1-\varepsilon,\ 8\varepsilon \leq \delta \leq 1. \tag{5}$$

Note that Equations (4) and (5) can always be satisfied by setting Φ to be large enough and f to be small enough. Also note that (4) and (5) are not tight bounds on the parameters and are just sufficient conditions for the analysis to work.

We now define what a typical execution, which will help us bound the random variables in our analysis.

Definition 6.10 (Typical Execution). *For any set S of at least ℓ consecutive good rounds, any set of J consecutive adversarial queries and $\alpha(J) = 2(\frac{1}{\varepsilon}+\frac{1}{3})\lambda/T(J)$, an execution E is typical if*

$$(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^{\Delta} pn(S) < Q^{i/P}(S) \leq D^{i/P}(S) < (1+\varepsilon)pn(S),$$
$$A^P(J) < p|J| + \max\{\varepsilon p|J|, \tau\alpha(J)\},$$
$$A^i(J) < p|J| + \max\{\varepsilon p|J|, b^i(J)\lambda(\frac{1}{\varepsilon}+\frac{1}{3})\},$$

where $b^i(J) = max_{j\in J} sup\{A_j^i - A_{j-1}^i | \mathcal{E}_{j-1} = E_{j-1}\}$.

We now show that a typical execution is a high-probability event.

Theorem 6.11. *For an execution $\mathcal{E}$ of $r_{\max}$ rounds, in a (γ,s)-respecting environment, the probability of the event "$\mathcal{E}$ not typical" is bounded by $O(r_{\max}^2)e^{-\lambda}$.*

The proof for Theorem 6.11 can be found in Appendix C.1

6.4 Non-pivot chain properties

Pivot chain behaves similar to the Bitcoin chain and its properties can be found in Appendix B

Next we prove some desired properties for the non-pivot chains.

Lemma 6.12 (Chain growth for non-pivot chain, from [13]). *Suppose that at round u of an execution E, an honest party broadcasts a i-th non-pivot chain of difficulty d. Then, by round v, every honest party receives a chain of difficulty at least $d + Q^i(S)$, where $S = \{r : u+\Delta \leq r \leq v-\Delta\}$.*

The proof of Lemma 6.12 is identical to Lemma B.3.

At round r, to mine on a non-pivot chain block, an honest party picks a target from the tip of a pivot chain in $\mathcal{S}_r^P$ which has good targets at round r because of Lemma B.4. So, as a consequence of **M1**, all the targets used by the honest parties on a non-pivot chain also satisfies $f/2\gamma^2 \leq pn_r T_r \leq f(1+\delta)\gamma^2$.

Lemma 6.13 (Common prefix for non-pivot chains). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, each non-pivot chain satisfies the common-prefix property with parameter $\ell_{\text{cp}} = \ell + 2\Delta$.*

The proof of Lemma 6.13 is in Appendix C

Lemma 6.14 (Chain quality for non-pivot chains). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, each non-pivot chain satisfies the chain-quality property with parameter $\ell_{\text{cq}} = \ell + 2\Delta$ and $\mu = \delta - 3\varepsilon$.*

The proof of Lemma 6.14 is in Appendix C.3.

6.5 Persistence and Liveness of Prism

Lemma 6.15 (Leader sequence common prefix). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, the leader sequence satisfies the leader-sequence-common-prefix property with parameter $\ell_{\text{lscp}} = 2\ell + 4\Delta$.*

Lemma 6.16 (Leader sequence quality). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, the leader sequence satisfies the leader-sequence-quality property with parameter $\ell_{\text{lsq}} = \ell + 2\Delta$ and $\mu = \delta - 3\varepsilon$.*

The proofs of Lemma 6.15 and Lemma 6.16 are in Appendix C.4.

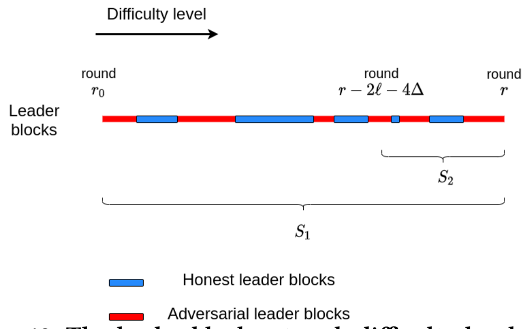


Figure 10: The leader blocks at each difficulty level in the proposer tree.

Theorem 6.17 (Persistence and liveness of Prism). *For a typical execution in a $(\gamma,2(1+\delta)\gamma^2\Phi/f)$-respecting environment,* Prism *satisfies persistence and liveness with parameter* $u = \frac{4(1+\varepsilon)\gamma^2(\ell+2\Delta)}{(\delta-3\varepsilon)(1-\varepsilon)^2}$.

Proof. By our definition, the persistence of Prism is equivalent to the leader sequence common prefix property proved in Lemma 6.15.

We next prove the liveness property. Suppose a transaction tx is received by all honest nodes before or at round r_0. Let $r \geq r_0 + u$ be current time and we shall prove that tx is contained in the permanent leader sequence of all honest nodes at round r. As shown in Figure 10, let $S_1 = \{r_0, \cdots, r\}$, $S_2 = \{r - 2\ell - 4\Delta, \cdots, r\}$, and J be the adversarial queries in S_2. By Lemma 6.15, for a difficulty level d, if d is covered by an honest block mined in $S1 \backslash S2$, then the block covering d will be permanent in the leader sequence at round r. We know that the difficulty level grows at least $Q^P(S_1) \geq (1-\varepsilon)^2 pn_r u/\gamma$ in S_1. By Lemma 6.16, we have that among the chain growth in S_1, different difficulty levels with size at least $(\delta-3\varepsilon)(1-\varepsilon)^2 pn_r u/\gamma$ is covered by honest leader blocks (which may not be permanent at round r). On the other hand, the proposer blocks that are not permanent (mined in S_2) cover different difficulty levels with size at most

$$\begin{aligned} D^P(S_2) + A^P(S_2) < 2D^P(S_2) \leq 2(1+\varepsilon)p\gamma n_r(2\ell+4\Delta) \\ = 4(1+\varepsilon)p\gamma n_r(\ell+2\Delta). \end{aligned}$$

Hence at least one honest proposer block B mined after r_0 is permanent in the leader sequence at round r. Since either B or some proposer block referred by B will contain tx, in both case we can conclude the proof.

□

7 EVALUATION

In our evaluation, we answer the following questions.

- Is the proposed scheme effective in matching the mining difficulty and the miner hash power?
- Does the blockchain forking rate remain low under our scheme, even with changing miner hash power?
- Does our scheme ensure that non-pivot chains adopt the difficulty of pivot chains, even with presence of the adversary?
- Does our scheme cause major computation and communication overhead when applied?

7.1 Experimental Setup

Simulator. To evaluate our scheme, we build a mining simulator for parallel-chain protocols in Golang. The simulator uses a round-by-round model with an adjustable round interval. In each round, blocks are mined on each of the parallel chains, and the number of blocks mined is determined by drawing from independent Poisson random variables with mean set to the product of the round interval and the per-chain mining rate. Miners receive newly-mined blocks after an adjustable network latency.

Simulated protocol. Our simulator does not consider the *interpretation* of the chains, such as transaction confirmation, ledger formation, etc. We only simulate the mining process. As a result, our evaluation is not tied to any particular protocol. Meanwhile, it is meaningful broadly to all PoW parallel-chain protocols, because they share this mining process.

There are 1 pivot chain and 1000 non-pivot chains. We simulate PoW mining on each of the chains at the same mining rate f. Each pivot-chain block contains its timestamp, difficulty, and parent. Each non-pivot-chain block also contains all these fields, plus a reference to a pivot-chain block (**M1**). We simulate two parties of miners: honest and adversary. Honest miners follow the general methodology described in section 1 by always referring to the best block in the pivot chain. They enforce the rules **M1**, **M2** by rejecting any non-compliant block . We design different adversarial miners to simulate attacks, and we provide more details later in Section 7.3.

Parameters. The round interval and the network latency are set to 2 seconds according to data collected in large-scale experiments of Prism [25]. The target mining rate f is set to 0.1 block per second per chain according to [25]. The epoch length Φ is set to 2016 blocks, and the dampening filter τ is set to 4 according to Bitcoin . We replay the historical Bitcoin mining power data [1] during the simulation.

7.2 Adaptation to Changing Miner Power

The main purpose of our scheme is to ensure the mining difficulty adapts to changing mining power. To show that, we simulate our scheme while varying the mining power according to the historical Bitcoin miner hash rate trace from Jan 2, 2019 to Feb 20, 2020. Figure 14 shows that even though the miner hash power has tripled during the simulated period, the mining difficulty of every chain keeps tracking the miner hash power very closely. Also, at any point in time, the max and min difficulty of all chains are very close. This demonstrates that the mining difficulty of all chains are always closely coupled, and no single chain experiences unstable difficulty or vulnerability.

As mentioned in Section 1, support for variable miner power is crucial to keeping the blockchain secure. If the miner hash power increases while the mining difficulty stays the same, the forking rate will increase due to decreased block inter-arrival time. To show our scheme is effective in keeping the blockchain secure, we compare the forking rate of two simulations: one using our scheme and one using a fixed mining difficulty. We use the same Bitcoin mining power data as in the previous experiment, and Figure 11 shows the results. Here, we report the forking rate as the ratio of the number of blocks not on the longest chain, to the number of blocks on the longest chain. If a fixed difficulty is used, the forking rate quickly increases as the miner power increases, to almost tripling towards the end of the

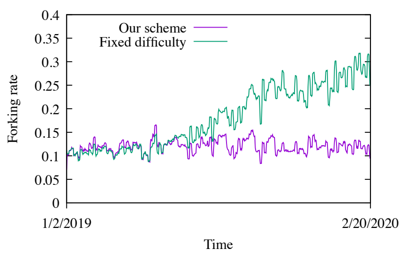


Figure 11: Forking rate of all parallel chains in two simulations, one using our scheme and one using fixed difficulty.

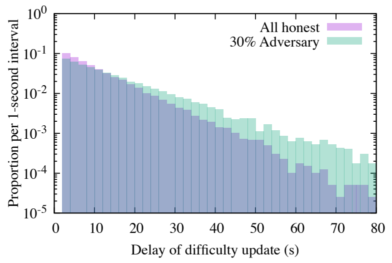


Figure 12: Frequency histogram of the delay where non-pivot chains update their difficulty to follow that of the pivot chain. Note the y-axis is log scale.

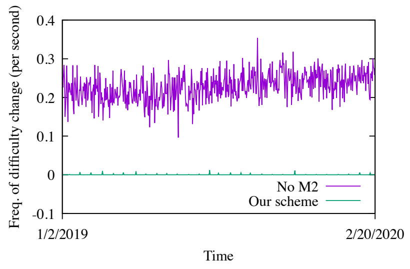


Figure 13: Frequency of difficulty change on a non-pivot chain where 30% of miner power is adversarial.

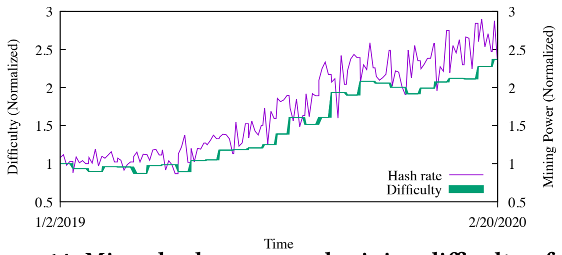


Figure 14: Miner hash power and mining difficulty of each chain when simulating our scheme over the historical Bitcoin **miner power trace. Difficulty is plotted as a region to show the max and min difficulty across all chains. Both metrics are normalized over their initial values.**

simulation. In comparison, our scheme keeps the forking rate low across all parallel chains for the whole simulation. This is because the mining difficulty and the miner hash power are closely matched under our scheme, so the block mining rate stays at a safe level.

7.3 Difficulty Update on Non-pivot Chains

One major challenge in designing our scheme is to ensure non-pivot chains adopt the pivot chain difficulty quickly after a new epoch begins, and we achieve it with the **M2** (Monotonicity, cf. Section 1). To show that adversarial miners cannot delay this process, we simulate our scheme where 30% of miners are adversarial. Adversarial miners do not voluntarily refer to the latest block on the pivot chain after a new epoch begins, but rather try to stay in the previous epoch (and mining difficulty) for as long as possible. We also simulate an all-honest scenario for comparison. We measure how soon non-pivot chains adopt new difficulty by tracking the delay from the last block of the previous epoch on the pivot chain to the first block of the new epoch of the non-pivot chain. Figure 12 shows the results. In either scenario, the difficulty of non-pivot chains is updated within 1–5 block intervals (0–50 seconds in real time). Although adversarial presence does delay the update of difficulty, the delay is not significant. This demonstrates that our mechanism ensures in-time update of non-pivot-chain difficulty.

We demonstrate that **M2** is essential to ensuring the mining difficulty does not vary too frequently on non-pivot chains. We compare two simulations where 30% of miner power is adversarial. In one case, we apply our full scheme. In the other case, we disable **M2** so that the adversary is free to choose whatever block on the pivot chain to refer when mining non-pivot chain blocks. Specifically, the adversary always tries to mine blocks with the lowest difficulty possible by referring to the genesis pivot-chain block. We focus on one non-pivot chain, and track the *frequency* of difficulty change. Difficulty change is defined as a block on the longest chain having different difficulty than its parent. Figure 13 shows the results. Under our scheme, non-pivot chain difficulty does not change for most of the time, and only changes swiftly at the beginning of new epochs, so the curve for our scheme stays close to zero. On the contrary, if we disable **M2**, the difficulty oscillates violently, as frequently as 0.2 times per second on average. This shows that our design is essential to maintain stable mining difficulty of non-pivot chains.

7.4 Analysis of Overhead

Finally, we analyze and show that our schemes will cause minimal overhead when implemented on existing parallel-chain protocols.
Communication and storage. Every block on the non-pivot chains needs to refer to a block on the pivot chain (**M1**), which takes the size of a hash (usually 32 bytes). This is a very small overhead compared to the size of the blockchain. For example, in Prism, the size of a voter (non-pivot-chain) block is 534 bytes [25]. The pivot-chain reference constitutes to an increase of 6% in communication and storage cost for voter blocks. Notice that voter blocks themselves only make up for 0.21% of the size of the Prism blockchain [25], so the overhead of pivot-chain referencing is negligible, regardless of the parameters.
Computation. Our scheme changes the mining and the transaction confirmation process of parallel chain protocols. For mining, notice that the pivot chain follows the same difficulty adjustment rule as Bitcoin, which is proven practical by its real-world deployment. Mining on non-pivot chains uses the same difficulty as the pivot chain, so there is no additional bookkeeping.

For transaction confirmation, we use Prism as a concrete example (note that no computation overhead exists in transaction confirmation for OHIE and FruitChains). Under static difficulty, Prism selects a leader for every *level* of the proposer tree. With **M3**, we partition the proposer tree into real-valued difficulty *intervals* such that no interval is *partially* occupied by any proposer block. We need to select a leader for each of such *intervals* (section 3.3). To determine the overhead, we need to answer: how many more intervals are there compared to levels?

We simulate the mining process of Prism with 1000 voter chains, epoch length $\Phi = 2016$ blocks, target mining rate $f = 0.1$ block per second, and found the number of intervals is only 0.12% more than

Table 2: Confirmation overhead vs epoch length Φ

Φ	10	100	1000	2016
Overhead	0.43%	0.07%	0.11%	0.12%

the number of levels. That is, our scheme incurs a confirmation overhead of 0.12%. This is expected, because only forks that happen at the beginning of an epoch will lead to extra intervals, and such a fork rarely exists with $\Phi = 2016$ and $f = 0.1$. Decreasing Φ may cause the overhead to increase because there are more epochs and it is more likely to fork at the beginning of an epoch. Table 2 plots the confirmation overhead for different Φ; we see that even at $\Phi = 10$, the overhead is smaller than 1%.

8 DISCUSSION

We presented a general methodology by which any parallel chain protocol can be converted from the fixed difficulty to the variable difficulty setting. We also proved the safety, liveness, and performance of the proposed scheme using novel proof method that analyzes the coupling between the pivot and non-pivot chains. There are several open directions of research. 1) In our design methodology, we proposed using a single chain as a pivot chain to set the difficulty target for all blocks. However, if we can use the information (for example, inter-block arrival times) from all the chains together to determine the difficulty target, we can get much better statistical averaging. This can lead to protocols which can adapt to much more aggressive mining power variation than is possible with a single-chain protocol. Such a protocol needs to be designed with care since it leads to strong coupling across all the chains. In particular, every chain needs to know the state of all other chains in order to check the correctness of the difficulty target. Since other chains can have forking in the meanwhile, it may lead to unintended complex interactions. 2) We analyzed various protocols under the variable difficulty setting. One new protocol, called Ledger-combiners [10] uses parallel-chains for robustly combining multiple ledgers as well as for achieving low latency. Analyzing that protocol in the variable difficulty setting is an interesting direction for future work.

9 ACKNOWLEDGEMENTS

This research is supported in part by a gift from IOHK Inc., an Army Research Office grant W911NF1810332 and by the National Science Foundation under grants CCF 17-05007 and CCF 19-00636.

REFERENCES

[1] Blockchain charts - total hash rate. https://www.blockchain.com/charts/hash-rate.

[2] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018.

[3] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019.

[4] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*, 2013.

[5] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy*, pages 104–121. IEEE, 2015.

[6] T-H. Hubert Chan, Naomi Ephraim, Antonio Marcedone, Andrew Morgan, Rafael Pass, and Elaine Shi. Blockchain with varying number of players. Cryptology ePrint Archive, Report 2020/677, 2020. https://eprint.iacr.org/2020/677.

[7] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.

[8] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms.* Cambridge University Press, 2009.

[9] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*, pages 45–59, 2016.

[10] Matthias Fitzi, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ledger combiners for fast settlement. In *Theory of Cryptography Conference*, pages 322–352. Springer, 2020.

[11] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[12] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[13] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. Full analysis of nakamoto consensus in bounded-delay networks. Cryptology ePrint Archive, Report 2020/277, 2020. https://eprint.iacr.org/2020/277.

[14] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.

[15] Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. Scaling nakamoto consensus to thousands of transactions per second. *arXiv preprint arXiv:1805.03870*, 2018.

[16] Songze Li and David Tse. Taiji: Longest chain availability with bft fast confirmation. *arXiv preprint arXiv:2011.11097*, 2020.

[17] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 2015.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008.

[19] Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, 2020.

[20] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[21] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2017.

[22] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

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

[24] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[25] Lei Yang, Vivek Bagaria, Gerui Wang, Mohammad Alizadeh, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Scaling bitcoin by 10,000 x. *arXiv preprint arXiv:1909.11261*, 2019.

[26] Haifeng Yu, Ivica Nikolić, Ruomu Hou, and Prateek Saxena. Ohie: Blockchain scaling made simple. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 90–105. IEEE, 2020.

APPENDIX

A THE DIFFICULTY RAISING ATTACK

Bitcoin set its target recalculation using a “dampening filter”-like adjustment (as defined in Definition 6.7). It turns out that this design is surprisingly foresighted. If we make a relaxation of the adjustment mechanism by removing the dampening filter, then it is subject to an attack called difficulty raising attack firstly discovered in [4]. At a high level, in this attack the adversary mines private blocks with timestamps in rapid succession, and induce one block with arbitrarily high difficulty in the private chain; via an anti-concentration argument, a sudden adversarial advance that can break agreement amongst honest parties cannot be ruled out. In this appendix, we describe this attack in detail and explain why having a “dampening filter” in the target recalculation function could resolve it.

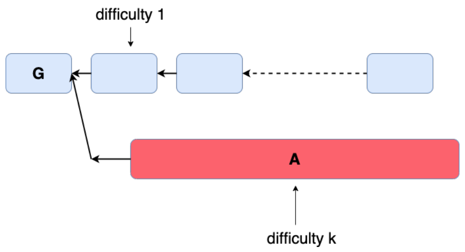


Figure 15: A simple attack if allowing miners to choose their own difficulty. The adversary mines one block which is as difficult as k honest blocks.

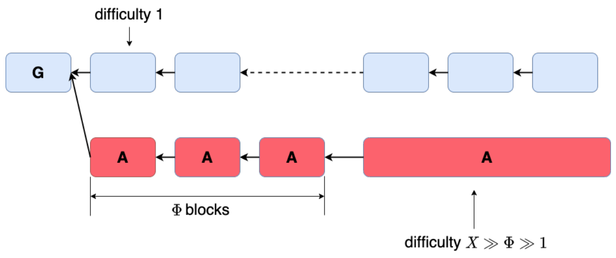


Figure 16: The difficulty raising attack. The adversary raises the difficulty to extremely high in the second epoch by faking timestamps.

A simple attack. As a prelim, we first look at a simple attack if the protocol lets miners to choose their own difficulty and use the heaviest chain rule. At a first glance, this rule appears kosher - the heaviest chain rule seems to afford no advantage to any miner to manipulate their difficulty. However, this lack of advantage only holds in expectation, and the variance created by extremely difficult adversarial blocks can thwart a confirmation rule that confirms deeply-embedded blocks, no matter how deep, with non-negligible probability. We give a simple calculation here. For simplicity, we using the difficulty defined in the genesis block as the difficulty unit and the expected inter-block time (10 minutes in Bitcoin) as the time unit. Let n be number of honest queries to the hash function per unit time and t be the number of adversarial queries per unit time. Then we know that to mine a block with unit difficulty, each query solves the PoW puzzle with probability $1/n$. We further assume that n and t don't change over time and the network delay among honest nodes is zero. Note that these assumptions only make the adversary weaker. The goal of the adversary is to double-spend a coin by mining a heavier chain than the public honest chain from the genesis.

Suppose honest miners are adopting the initial mining difficulty as defined in the genesis block, hence on average it take k units of time to mine a honest chain with k blocks. To mine a heavier chain, the adversary only needs to mine one block which has difficulty k (See Figure 15 for illustration), within k unit of time. The adversarial can make tk queries in k units of time, and each query succeeds with probability $1/nk$. Hence the success probability of this attack would be

$$\mathbb{P}(\text{attack succeeds}) = 1-(1-\frac{1}{nk})^{tk} \approx 1-e^{t/n},$$

since n and t are large in PoW mining. Note that the success probability is a constant independent of k, therefore any k-deep confirmation rule will fail.

Difficulty raising attack. However, even if we adopts a epoch based difficulty adjustment rule as in Bitcoin (but without the "dampening filter"), there is still a difficulty raising attack. We using the difficulty of the first epoch (defined in the genesis block) as the difficulty unit and the expected inter-block time (10 minutes in Bitcoin) as the time unit. Let Φ be the length of an epoch in number of blocks (2016 in Bitcoin). And we define n and t the same as above.

Note that the adversary can put any timestamp in its private blocks, so the difficulty of the second epoch in its private chain can be arbitrary value as long as the adversary completes the first epoch. Let B with difficulty X be the first block of the second epoch in the private chain (that is each query solves the PoW puzzle with probability $1/nX$), then B has chain difficulty $\Phi+X$. See Figure 16 for illustration. To mine an honest chain with chain difficulty $\Phi+X$, on average it takes $\Phi+X$ time. On the other hand, it takes on average $n\Phi/t$ time for the adversary to complete the first epoch in its private chain. Therefore, to succeed in this attack, the adversary needs to mine the block B within $\Phi+X-n\Phi/t$ time, which happens with probability:

$$\begin{aligned}\mathbb{P}(\text{attack succeeds}) &= 1-(1-\frac{1}{nX})^{(\Phi+X-n\Phi/t)t}\\ &= 1-(1-\frac{1}{nX})^{Xt-(n-t)\Phi}\\ &\approx 1-e^{t/n},\end{aligned}$$

if $X \gg \Phi \gg 1$. Note that the success probability is independent of the length of the public longest chain, hence any k-deep confirmation rule will fail.

However, Bitcoin is saved by the dampening filter in the target recalculation function. As in Definition 6.7, the difficulty can be increased by a factor of at most τ between two consecutive epochs ($\tau = 4$ in Bitcoin). Then we shall analyze the difficulty raising attack under the same assumptions made above. Since the epoch size $\Phi \gg 1$, the time for the adversary to complete one epoch or mine Φ blocks with the same difficulty will satisfy the concentration bound of binomial random variables. Hence if the adversary always rises the difficulty by τ in each epoch, then it takes on average $\frac{n}{t}\sum_{i=0}^{\ell-1}\tau^i\Phi$ time for the adversary to complete ℓ epochs in its private chain, and the public honest chain will on average have difficulty $\frac{n}{t}\sum_{i=0}^{\ell-1}\tau^i\Phi$ during this time. Since the private chain has chain difficulty $\sum_{i=0}^{\ell-1}\tau^i\Phi$, the gap of chain difficulties between the public honest chain and the private chain will be

$$(\frac{n}{t}-1)\sum_{i=0}^{\ell-1}\tau^i\Phi = (\frac{n}{t}-1)\frac{\tau^\ell-1}{\tau-1}\Phi.$$

Each block of the $(\ell+1)$-th epoch in the private chain will have difficulty τ^ℓ, hence the adversary still needs to mine approximately $\frac{n-t}{t(\tau-1)}\Phi$ blocks in order to catch up the honest chain. As $\Phi \gg 1$, the time for the adversary to catch up is still controlled by the concentration bound, and the success probability of this attack will be at most $e^{-\theta(\Phi)}$. By setting Φ large enough, the difficulty raising attack can be ruled out.

While this specific attack could in principle be thwarted, to have security guarantee we still need to consider all possible attacks in the presence of a full-blown adversary. A full and beautiful analysis of Bitcoin rule is provided in [13] and we shall give a proof sketch in Appendix B.

B BITCOIN BACKBONE PROPERTIES REVISITED

We will briefly revisit the analysis in [13] because the pivot chain is identical to the Bitcoin chain.

We will additionally define a stale chain and accuracy related to timestamps of the blocks.

Definition B.1 (from [13]). *A block created at round u is accurate is it has a timestamp v such that $|u-v| \leq \ell+2\Delta$. A chain is accurate if all its blocks are accurate. A chain is stale if for some $u \geq \ell+2\Delta$ it does not contain a honest block with timestamp $v \geq u-\ell-2\Delta$.*

Recall that we define $\mathcal{S}_r^P$ as the set of pivot chains that belong to or have the potential to be adopted by an honest party at round r in Section 6.3. Now we define a series of useful predicates with respect to $\mathcal{S}_r^P$.

Definition B.2 (from [13]). *For a round r,*
GoodRounds(r):= *"All rounds $u \leq r$ are good."*
GoodChains(r):= *"For all rounds $u \leq r$, every chain in $\mathcal{S}_u^P$ is good."*
NoStaleChains(r) := *"For all rounds $u \leq r$, there is no stale chain in $\mathcal{S}_u^P$."*
Accurate(r) := *"For all rounds $u \leq r$, all chains in $\mathcal{S}_u^P$ are accurate."*
Duration(r):= *"For all rounds $u \leq r$ and duration Λ of an epoch of any chain in $\mathcal{S}_u^P$, $\frac{1}{2(1+\delta)\gamma^2}\frac{m}{f} \leq \Lambda \leq 2(1+\delta)\gamma^2\frac{m}{f}$."*

The following lemma provides a lower bound on the progress of the honest parties, which holds irrespective of any adversary.

Lemma B.3 (Chain growth for pivot chain, from [13]). *Suppose that at round u of an execution E, an honest party broadcasts a pivot chain of difficulty d. Then, by round v, every honest party receives a chain of difficulty at least $d+Q^P(S)$, where $S=\{r: u+\Delta \leq r \leq v-\Delta\}$.*

In order to prove properties like common prefix and chain quality for the pivot chain, we need all rounds in a typical execution to be good.

Lemma B.4 (All rounds in a typical execution are good, Theorem 2 from [13]). *Consider a typical execution in a $(\gamma, 2\gamma^2(1+\delta)\Phi/f)$-respecting environment. If the protocol is initiated such that the first round it good, and all the conditions 3, 4 and 5 are satisfied, then all rounds are good.*

Proof. The elaborate proof can be found in [13] and we summarize it as follows.

We will use an induction argument. In a (γ,s)-respecting environment, $s \geq 2(1+\delta)\gamma^2 m/f$ covers at least the first epoch. It is easy to see that if the initial target is good, the rounds in the first epoch are good, and the first target recalculation point is good. We will prove that the subsequent rounds and target recalculation points are good using an induction argument shown in Figure 17. The predicates are defined as follows.

We prove *NoStaleChains*(r) from *GoodRounds*$(r-1)$ using typicality bounds, showing that the adversary cannot accumulate more

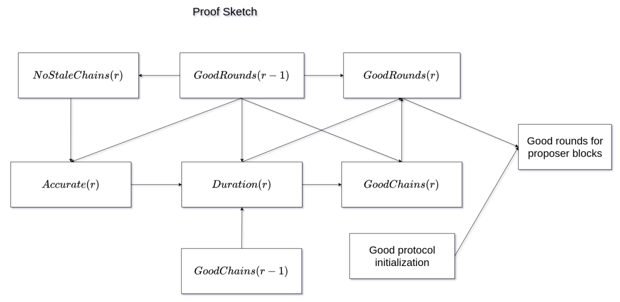


Figure 17: An induction argument to prove that all rounds in a typical execution are good.

difficulty than the lower bound of the minimum chain growth, Q^P. Let w be the timestamp of the last honest block on the stale chain. Set $U=\{u: w \leq u \leq r\}$, $S=\{u: w+\Delta \leq u \leq r-\Delta\}$ and J be the adversarial queries in U. We will first consider the case where the chain has more than one target recalculation point. In this case we divide J into sub-queries J_i such that each subset covers at least $m/2$ blocks and has exactly one target recalculation point in it. In this case, we have $A^P(J) = \sum_i A^P(J_i) < \sum_i (1+\varepsilon)p|J_i| = (1+\varepsilon)p|J|$. We arrive at a contradiction by showing $(1+\varepsilon)p|J|$ is lower than $Q^P(S)$'s lower bound. In case of at most one target recalculation point, if $A(J) < (1+\varepsilon)p|J|$ applies, the argument from the previous case applies. If $A(J) < (1+1/\varepsilon)(1/3+1/\varepsilon)\lambda\tau/T(J)$, we prove that the lower bound of $Q^P(S)$ considering only the first l rounds will cover $(1+1/\varepsilon)(1/3+1/\varepsilon)\lambda\tau/T(J)$. *Accurate*$(r)$ follows from *NoStaleChains*(r).

We then prove the bound on duration by contradiction, assuming that the previous target recalculation point is good using property *GoodChains*$(r-1)$. The lower bound is contradicted by showing that even if the adversary and honest party join forces, they can't produce e blocks in less than $\frac{1}{2(1+\delta)\gamma^2}\frac{m}{f}$. The upper bound is contradicted by showing that the lower bound of Q^P produces at least e blocks in $2(1+\delta)\gamma^2\frac{m}{f}$ rounds. To prove *GoodChains*(r), we prove that the next target-recalculation point is good. This is proved again, using a contradiction for both the bounds of a good target recalculation point. Finally, we use *Duration*(r), *GoodChains*(r) and the $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment assumption to prove *GoodRounds*(r). ◻

The following lemma from [13] is useful to prove common prefix and chain quality of the pivot chain.

Lemma B.5 (Lemma 2(c) from [13]). *Consider a typical execution in a (γ,s)-respecting environment. Let $S=\{r: u \leq r \leq v\}$ be a set of rounds with at least ℓ rounds and J be the set of adversary queries in $U=\{r: u-\Delta \leq r \leq v+\Delta\}$. If w is a good round such that $|w-r| \leq s$ for any $r \in S$, then $A^P(J) < (1-\delta+3\varepsilon)Q^P(S)$*

The following properties of the pivot chain are from [13].

Lemma B.6 (Common prefix for pivot chain). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, the pivot chain satisfies the common-prefix property with parameter $\ell_{\text{cp}} = \ell+2\Delta$.*

Lemma B.7 (Chain quality for pivot chain). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, the pivot chain satisfies the chain-quality property with parameter $\ell_{\text{cq}} = \ell+2\Delta$ and $\mu = \delta-3\varepsilon$.*

C PROOF FOR SECTION 6

C.1 Proof for typical execution

The following concentration bound on a martingale is helpful to bound the probability of a not typical execution.

Theorem C.1 (from [13]). *Let $(X_1,X_2,...)$ be a martingale with respective the sequence $(Y_1,Y_2,...)$, if an event G implies $X_k - X_{k-1} \le b$ and $V = \sum_k var[X_k - X_{k-1}|Y_1,...,Y_{k-1}] \le v$, then for non-negative n and t*

$$P(X_n - X_0 \ge t, G) \le e^{-\frac{t^2}{2v+\frac{2bt}{3}}}.$$

And for the proof of Theorem 6.11

Proof. The proof for $Q^P(S), D^P(S)$ and $A^P(J)$ can be found in [13] and the same proof follows for $Q^i(S)$ and $D^i(S)$. We will prove it for $A^i(J)$. For each $j \in J$, let A_j be the difficulty of the block obtained with the j^{th} query as long as the target was at least $1/b^i(J)$. Define

$$X_0 = 0,$$
$$X_k = \sum_{j\in[k]} A_j - \sum_{j\in[k]} \mathbb{E}[A_j|\mathcal{E}_{j-1}], k \in [|J|],$$

which is a martingale with respect to the sequence $\mathcal{E}_{j-1}, j \in J$. For the above martingale, for all $k \in [|J|]$, we have $X_k - X_{k-1} \le b^i(J)$, using the definition of $b^i(J)$ and $var[X_k - X_{k-1}] \le pb^i(J)$ and $\mathbb{E}[A_j|\mathcal{E}_{j-1}] \le p$. We will apply Theorem C.1 with $t = \max\{\epsilon p|J|, b^i(J)\lambda(\frac{1}{\epsilon}+\frac{1}{3})\} \ge b^i(J)\lambda(\frac{1}{\epsilon}+\frac{1}{3})$ and $v = b^i(J)p|J|$ to obtain

$$Pr[\sum_{j\in J} A_j \ge p|J| + t] \le exp\{-\frac{t}{2b^i(J)(\frac{1}{3}+\frac{1}{\epsilon})}\} \le e^{-\lambda}.$$

□

Lemma C.2 (Proposition 2 from [13]). *In a (γ,s)-respecting environment, let U be a set of at most s consecutive rounds and $S \subseteq U$ then, for any $n \in \{n_r : r \in U\}$ we have*

$$\frac{n}{\gamma} \le \frac{n(S)}{|S|} \le \gamma n,$$
$$n(U) \le (1 + \frac{\gamma|U\setminus S|}{|S|})n(S).$$

C.2 Proof of Lemma 6.13

By the definition of typical execution, we have the following lemma that will be useful in the proof.

Lemma C.3. *Under a typical execution, for the set of rounds S with $|S| \ge \ell$, let $Q^P(S)$ correspond to the pivot tree and $Q^i(S)$ correspond to any non-pivot tree then, $Q^i(S) > Q^P(S)(1-\epsilon)[1-(1+\delta)\gamma^2 f]^\Delta/(1+\epsilon)$.*

Proof. This follows from the definition of typicality, we use the following inequalities

$$(1+\epsilon)pn(S) > Q^P(S),$$
$$Q^i(S) > (1-\epsilon)[1-(1+\delta)\gamma^2 f]^\Delta pn(S).$$

□

The following proposition will be useful in the proof of non-pivot chain's common prefix.

Proposition 1. *In a typical execution, we have the following bound*

$$A^i(J) < (1+\epsilon)p|J|$$

for $p|J| \ge \frac{2b^i(J)\lambda}{\epsilon}(\frac{1}{3}+\frac{1}{\epsilon})$.

$$A^i(J) < (1+\epsilon)\frac{2b^i(J)\lambda}{\epsilon}(\frac{1}{3}+\frac{1}{\epsilon}) < \frac{(1-\epsilon^2)(\frac{\epsilon}{3}+1)\epsilon\Phi}{8\gamma^5(1+\delta)(1+3\epsilon)}\frac{b^i(J)}{\tau}$$

for $p|J| < \frac{2b^i(J)\lambda}{\epsilon}(\frac{1}{3}+\frac{1}{\epsilon})$, the second inequality follows from the bound on ℓ.

For the proof of Lemma 6.13

Proof. Consider the i^{th} non-pivot chain, suppose common prefix fails for two chains C_1 and C_2 held by honest players at rounds $r_1 \le r_2$ respectively, that is, $\exists B \in C_1^{\lceil \ell+2\Delta}$, s.t. $B \notin C_2$. It is not hard to see that in such a case there was a round $r \le r_2$ and two honest held chains C and C' in $\mathcal{S}_r^i$, such that $B \in C^{\lceil \ell+2\Delta}$ but $B \notin C'$. Then we know B is a descendant of head($C \cap C'$), and hence head($C \cap C'$) $\in C^{\lceil \ell+2\Delta}$. Therefore, the timestamp of head($C \cap C'$) is less than $r - \ell - 2\Delta$.

Let $v < r - \ell - 2\Delta$ be the timestamp of head($C \cap C'$) and $w \le v$ be the timestamp of the last honest block B_h^i on ($C \cap C'$). Let $U^i = \{u : w \le u \le r\}, S^i = \{u : w + \Delta \le u \le r - \Delta\}$ and let J^i be the adversarial queries in rounds U^i. Let $\mathcal{S}^P_{r,w-\Delta}$ be the collection of pivot chains heavier than at least one chain in $\mathcal{S}_{w-\Delta}$. And for $j \in J^i$, let $\mathcal{S}^P_{j,w-\Delta}$ be the collection of pivot chains heavier than at least one chain in $\mathcal{S}_{w-\Delta}$. Due to condition **M2**, all the difficulties of the blocks in C or C' that come after B_0^i are extending $\mathcal{S}^P_{r,w-\Delta}$. We have $b = b^i(J) = \max_{j\in J^i}\sup\{A^i_j - A^i_{j-1}|\mathcal{E}_{j-1} = E_{j-1}\} = \max_{j\in J^i} sup\{A^i_j - A^i_{j-1}|\mathcal{E}_{j-1} = E_{j-1}\} = \max_{j\in J^i}\sup\{\text{diff}(C^P B^*)|C^* \in \mathcal{S}_{j,w-\Delta}\} = \sup_{C^P\in\mathcal{S}_{r,w-\Delta}}\{\text{diff}(C^P B^*)\}$. The last equality applies because, for $j \in J^i$, $\mathcal{S}_{j,w-\Delta} \subseteq \mathcal{S}_{r,w-\Delta}$. Let $C^* \in \mathcal{S}_{r,w-\Delta}$ be the chain for which diff$(C^* B^*) = b$. In case such C^* doesn't exist, there exists a sequence of chains C_n^*, such that diff$(C_n^* B^*)$ approaches b in limit. Let the block B_h^P be the last honest block on C^* with timestamp x.

We claim that if $r > \ell + 2\Delta + w$, then $A^i(J^i) < (1+\delta+3\epsilon)Q^i(S^i)$. The proof is as follows. When $p|J^i| \ge \frac{2b\lambda}{\epsilon}(\frac{1}{3}+\frac{1}{\epsilon})$, the concentration bound $A^i(J^i) < (1+\epsilon)p|J^i|$ applies. We have $n(U^i) \le n(S^i)(1+\gamma|U \setminus S|/|S|) < (1+\epsilon^2/2)n(S)$ and

$$A^i(J^i) < (1+\epsilon)(1-\delta)pn(U^i) < (1+\epsilon)(1+\epsilon^2/2)(1-\delta)pn(S^i)$$
$$< (1-\delta+\epsilon)pn(S^i) < (1-\delta+3\epsilon)Q^i(S^i)$$

We will prove this when $p|J^i| < \frac{2b\lambda}{\epsilon}(\frac{1}{3}+\frac{1}{\epsilon})$.

Case 1	C^* has at most one target-recalculation point after B_h^P and $w \le x \le r \Rightarrow f/2\gamma^2\tau < \frac{1}{b}n_x p$
Case 2	C^* has at least two target-recalculation point after B_h^P and $w \le x \le r$
Case 3	$x < w, \lvert w-x \rvert > \gamma^2(1+\delta)\Phi/f - \ell - 2\Delta$
Case 4	$x < w, \lvert w-x \rvert < \gamma^2(1+\delta)\Phi/f - \ell - 2\Delta$, and C^* has at most one target-recalculation point after B_h^P
Case 5	$x < w, \lvert w-x \rvert < \gamma^2(1+\delta)\Phi/f - \ell - 2\Delta$, and C^* has at least two target-recalculation point after B_h^P

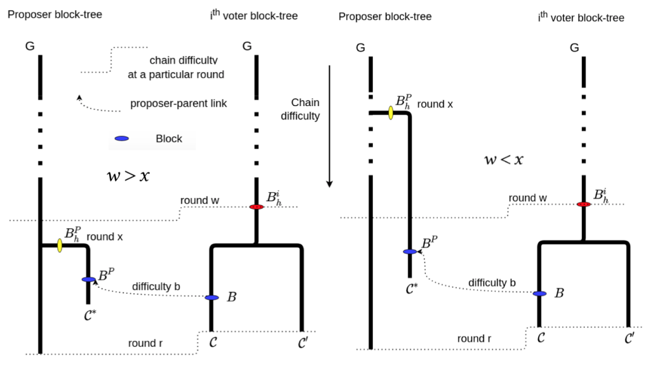


Figure 18: Common Prefix Proof (Left): $w < x$, **(Right):** $w > x$

Cases 1, 2 are shown in left and cases 3,4,5 in right of Figure 18.
Case 1: The last honest block B_h^P in the chain C^* has a timestamp $x \geq w$. We will look at the case when C^* has at most one target recalculation point after B_h^P. In this case the difficulty b satisfies $\frac{f}{2\gamma^2\tau} < \frac{1}{b} n_x p$ since the difficulty can raise by at most a factor of τ and considering the first ℓ rounds in S^i, we have $n(S^i) > \frac{n_x}{\gamma}\ell$. Using typicality we have, $p|J| \leq (1-\delta+\varepsilon^2/2)pn(S)$ and

$$\varepsilon(1-2\varepsilon)pn(S^i) > \varepsilon(1-2\varepsilon)\frac{pn_x \ell b}{\gamma b} > \frac{\varepsilon(1-2\varepsilon)f\ell b}{2\gamma^3\tau} \geq 2b\lambda(\frac{1}{\varepsilon}+\frac{1}{3}),$$
$$A^i(J^i) < p|J| + 2b\lambda(\frac{1}{\varepsilon}+\frac{1}{3}) \leq (1-\delta+\varepsilon)pn(S^i) < (1-\delta+3\varepsilon)Q^i(S^i)$$

Case 2: The last honest block B_h^P in the chain C^* has a timestamp $x \geq w$. We will look at the case when C^* has more than one target recalculation point after B_h^P. Let U^P be the set of rounds $\{u : x \leq u \leq r\}$, S^P be the set of rounds $\{u : x+\Delta \leq u \leq r-\Delta\}$ and J^P be the queries made by the adversary for the proposer chain in U^P. In this case difficulty accumulated by the adversary in J^P queries is at least $\frac{b}{\tau}\Phi$. Using typicality, we have $p|J^P| > \frac{b\Phi}{\tau(1+\varepsilon)}$ and using honest party's advantage we have $n(S^i) \geq n(S^P) > \frac{|J^P|}{(1-\delta)(1+\varepsilon^2/2)}$.

$$(1-\delta+3\varepsilon)Q^i(S^i) > (1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta \frac{(1-\delta+3\varepsilon)p|J^P|}{(1-\delta)(1+\varepsilon^2/2)}$$
$$> \frac{b\Phi}{\tau} \geq A^i(J^i).$$

The last inequality implies from Proposition 1.
Case 3: Consider the case when $x < w$ and $|w-x| > s/2-\ell-2\Delta$. Let $S' := \{u : x+\Delta \leq u \leq w-\Delta\}$, $U^P = \{u : x \leq u \leq r\}$ and J^P be the set of adversarial queries for the proposal tree in the rounds U^P. The difficulty accumulated in the chain C^* in J^P queries is more than that of the chain growth in S'.

$$A^P(J^P) \geq ChainGrowth^P(S') \geq Q^P(S').$$

Since $s = \frac{2\gamma^2(1+\delta)\Phi}{f}$, we have $|S'| \geq (1+\delta)(1-\varepsilon)\gamma^2\Phi/f$. Considering the first $s/2-\ell$ rounds in $U^P \setminus U^i$, if T_x is the target used by the honest party in round x, then $\frac{n(S')}{|S'|} \geq \frac{n_x}{\gamma}$ and $T_x n_x p \geq \frac{f}{2\gamma^2}$. Using these, we have

$$Q^P(S') > (1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta pn(S')$$
$$\geq (1+\delta)(1-\varepsilon)^3 \frac{\gamma\Phi pn_x T_x}{2fT_x} \geq (1+\delta)(1-\varepsilon)^3 \frac{\Phi}{2\gamma T_x} > \frac{\Phi}{2T_x\gamma}$$

Note that starting with target T_x, if C^* has at most one target recalculation point after B_h^P, then the accumulated difficulty is at least $\frac{\Phi}{2\gamma}\frac{b}{\tau}$, which is a smaller quantity than $\frac{\Phi}{2T_x\gamma}$. If the chain has more than one target recalculation point, then the accumulated difficulty is at least $m\frac{b}{\tau}$ which is larger than $\frac{\Phi}{2\gamma}\frac{b}{\tau}$. Hence, the accumulated difficulty will be at least $\frac{\Phi}{2\gamma}\frac{b}{\tau}$ in any case.

$$|J^P|p(1+\varepsilon) > A^P(J^P) \geq Q^P(S'),$$
$$A^P(J^P) > \frac{\Phi}{2\gamma}\frac{b}{\tau}$$

We have $n(S^i)+n(S') > \frac{|J^P|}{(1-\delta)(1+\varepsilon^2/2)}$ and

$$Q^i(S^i)+Q^i(S') > (1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta \frac{p|J^P|}{(1-\delta)(1+\varepsilon^2/2)},$$
$$Q^i(S') < Q^P(S')\frac{(1+\varepsilon)}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta}$$
$$< \frac{(1+\varepsilon)^2}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta}p|J^P|$$

Combining both we have,

$$Q^i(S^i) > p|J^P|\frac{((1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta)^2-(1+\varepsilon)^2(1-\delta)(1+\varepsilon^2/2)}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta(1-\delta)(1+\varepsilon^2/2)},$$

and then

$$(1-\delta+3\varepsilon)Q^i(S^i)$$
$$> (1+3\frac{\varepsilon}{1-\delta})\frac{(\delta(1+\varepsilon^2/2)(1+\varepsilon)^2-6\varepsilon)}{(1-\varepsilon^2)[1-(1+\delta)\gamma^2 f]^\Delta(1+\varepsilon^2/2)}\frac{\Phi}{2\gamma}\frac{b}{\tau}$$
$$> A^i(J^i),$$

Where the last inequality follows from the condition $\delta > 8\varepsilon$.
Case 4: Consider the case the last honest block in the chain containing B's proposer parent has a timestamp $x < w$ and $|w-x| < s/2-\ell-2\Delta$ and C^* has at most one target recalculation point after B_h^P. Let $S' := \{u : x+\Delta \leq u \leq w-\Delta\}$. The difficulty accumulated C^* in J^P queries is more than that of the chain growth in S'. Considering just first ℓ rounds in S^i, we have $n(S^i) > \ell n_x/\gamma$ and b satisfies $\frac{f}{2\gamma^2\tau} < \frac{1}{b}n_x p$. Using these bounds and Lemma B.3, we have

$$\varepsilon(1-2\varepsilon)pn(S^i) > \varepsilon(1-2\varepsilon)\frac{pn_x \ell b}{\gamma b} > \frac{\varepsilon(1-2\varepsilon)f\ell b}{2\gamma^3\tau} \geq 2b\lambda(\frac{1}{\varepsilon}+\frac{1}{3}),$$
$$A^i(J^i) < p|J| + 2b\lambda(\frac{1}{\varepsilon}+\frac{1}{3}) \leq (1-\delta+\varepsilon)pn(S^i) < (1-\delta+3\varepsilon)Q^i(S^i)$$

Case 5: Consider the case the last honest block in the chain containing B's proposer parent has a timestamp $x < w$ and $|w-x| < s/2-\ell-2\Delta$. Let $S' := \{u : x+\Delta \leq u \leq w-\Delta\}$. The difficulty accumulated by C^* in J^P queries is more than that of the chain growth in S'. We will consider the case C^* has more than one target recalculation point after B_h^P. The adversary accumulates more than $\frac{b}{\tau}\Phi$ difficulty in J^P queries and similar to **Case 4**, we have

$$|J^P|p(1+\varepsilon) > A^P(J^P) \geq \Phi\frac{b}{\tau},$$
$$A^P(J^P) \geq Q^P(S'),$$
$$n(S^i)+n(S') > \frac{|J^P|}{(1-\delta)(1+\varepsilon^2/2)},$$

and

$$Q^i(S^i)+Q^i(S') > (1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta \frac{p|J^P|}{(1-\delta)(1+\varepsilon^2/2)},$$
$$Q^i(S') < Q^P(S')\frac{(1+\varepsilon)}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta}$$
$$< \frac{(1+\varepsilon)^2}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta} p|J^P|$$

Combining both we have

$$Q^i(S^i) > p|J^P|\frac{((1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta)^2-(1+\varepsilon)^2(1-\delta)(1+\varepsilon^2/2)}{(1-\varepsilon)[1-(1+\delta)\gamma^2 f]^\Delta(1-\delta)(1+\varepsilon^2/2)},$$

and then

$$(1-\delta+3\varepsilon)Q^i(S^i)$$
$$>(1+3\frac{\varepsilon}{1-\delta})\frac{(\delta(1+\varepsilon^2/2)(1+\varepsilon)^2-6\varepsilon)\Phi b}{(1-\varepsilon^2)[1-(1+\delta)\gamma^2 f]^\Delta(1+\varepsilon^2/2)\tau}$$
$$>A^i(J^i),$$

The last inequality follows from the condition $\delta > 8\varepsilon$.

We also claim that, if $r-w > \ell+2\Delta$, then $2Q^i(S^i) \leq D^i(U^i)+A^i(J^i)$, which leads to a contradiction as $D^i(U^i) < (1+5\varepsilon)Q^i(S)$ and $A^i(J^i) < (1-\delta+3\varepsilon)Q^i(S^i)$.

Towards proving the claim above, associate with each $r \in S$ such that $Q_r^i > 0$ an arbitrary honest block that is computed at round r for difficulty Q_r^i. Let $\mathcal{B}$ be the set of these blocks and note that their difficulties sum to $Q^i(S)$. Then consider a block $B \in \mathcal{B}$ extending a chain C^* and let $d = \text{diff}(C^*B)$. If $d \leq \text{diff}(C \cap C')$ (note that $u < v$ in this case and head$(C \cap C')$ is adversarial), let B_0 be the block in $C \cap C'$ containing d. Such a block clearly exists and has a timestamp greater than u. Furthermore, $B_0 \notin \mathcal{B}$, since B_0 was an adversarial block. If $d > \text{diff}(C \cap C')$, note that there is a unique $B \in \mathcal{B}$ such that $d \in B$. Since B cannot simultaneously be on chain C and C', there is a $B_0 \notin \mathcal{B}$ either on C or on C' that contains d. Hence there exists a set of blocks $\mathcal{B}'$ computed in U such that $\mathcal{B} \cap \mathcal{B}' =$ and $\{d \in B : B \in \mathcal{B}\} \subseteq \{d \in B : B \in \mathcal{B}'\}$. Because each block in $\mathcal{B}'$ contributes either to $D^i(U)-Q^i(S)$ or to $A^i(J)$, we have $Q^i(S^i) \leq D^i(U^i)-Q^i(S)+A^i(J^i)$. □

C.3 Chain Quality of Non-pivot Chains

Proof of Lemma 6.14. Without loss of generality, we focus on the first non-pivot chain. Let B_i denote the i-th block of C and consider K consecutive blocks $B_u, \cdots, B_v$ in C with timestamp in S_0. Define K_0 as the least number of consecutive blocks $B_{u'},\cdots,B_{v'}$ that include the K given ones (i.e., $u' \leq u$ and $v \leq v'$) and have the properties (1) that the block $B_{u'}$ was mined by an honest party at some round r_1 or is the genesis block in case such block does not exist, and (2) that there exists a round r_2 such that the chain ending at block $B_{v'}$ is adopted by some honest node at round r_2. Let d' be the total difficulty of these K' blocks. Define $U = \{r_1,\cdots,r_2\}$, $S = \{r_1+\Delta,\cdots,r_2-\Delta\}$, and J the adversarial queries in U associated with the K' blocks. Then we have $|S| = |U|-2\Delta \geq |S_0|-2\Delta \geq \ell$. Then following the same argument from Lemma 6.13, we have $A^1(J) < (1-\delta+3\varepsilon)Q^1(S)$. Let x denote the total difficulty of all the blocks from honest parties that are included in the K blocks and—towards a contradiction—assume $x < \mu d \leq \mu d'$. In a typical execution, all the K' blocks have been mined in U. But then we have the following contradiction

$$A^1(J) \geq d'-x > (1-\mu)d' \geq (1-\mu)Q^1(S) = (1-\delta+3\varepsilon)Q^1(S).$$

Therefore, we can conclude the proof. □

C.4 Common Prefix and Chain Quality of the Leader Sequence

Proof of Lemma 6.15. Let $r \geq R_d+2\ell+4\Delta$ be the current round. For $1 \leq i \leq m$, let C_i be the heaviest voter chain i in an honest node u's view at round r. By the common prefix property in Lemma 6.13, blocks in $C_i^{\lceil \ell+2\Delta}$ remain unchanged until $r_{\max}$. In addition, by the chain quality property in Lemma 6.14, we know that for $1 \leq i \leq m$, there exists at least one honest block B_i on chain C_i whose timestamp is in the interval $(r-2\ell-4\Delta, r-\ell-2\Delta)$,i.e., B_i is on the chain $C_i^{\lceil \ell+2\Delta}$. As B_i is an honest block mined after R_d, B_i or an ancestor of B_i must have voted for the difficulty level d. Therefore the leader sequence remains unchanged up to difficulty level d until $r_{\max}$. □

Proof of Lemma 6.16. Let r be the current round, C be the proposer chain held by honest player P, and $d = \text{diff}(C)$. Let interval $D = (d',d]$ be the difficulty range covered by all blocks in C with timestamp in last $\ell+2\Delta$ rounds. Define:

$$d^* := \max(\tilde{d} \leq d' \ s.t \ \text{ the honest players mined}$$
$$\text{the first proposer block covering } \tilde{d})$$

Let r^* be the round in which the first proposer block covering d^* was mined. $r^* = 0$ and $d^* = 0$ if such proposer block does not exists. Define $U = \{r^*,\cdots,r\}$, $S = \{r^*+\Delta,\cdots,r-\Delta\}$, and J the adversarial queries in U. Then we have $|S| = |U|-2\Delta \geq \ell$. From the definition of d^* we have the following two observations:

(1) All difficulties in $(d^*,d']$ are covered by at least one adversarial proposer block.
(2) All the proposer blocks covering $(d^*,d]$ are mined in the interval U because there are no proposer blocks covering d^* before round r^* and hence no player can mine a proposer block covering a difficulty level greater than d^* before round r^*.

Let L_h be the size of difficulty range covered by honest leader blocks in the range $(d',d]$ and say

$$L_h < \mu(d-d') \leq \mu(d-d^*). \tag{6}$$

Let L_h' be the size of difficulty range covered by honest leader blocks in the range $(d^*, d']$. The adversarial leader blocks have covered difficulty ranges with size $d-d^*-L_h-L_h'$ in the interval U. From our first observation, we know that adversarial proposer blocks in the difficulty range $[d^*,d']$ which are *not* leader blocks cover difficulty ranges with size at least L_h', and from our second observation, these proposer blocks are mined in the interval U. Therefore, we have the following bound on $A^P(J)$

$$\begin{aligned} A^P(J) &\geq (d-d^*-L_h-L'_h)+L'_h \\ &= d-d^*-L_h \\ \text{(From Equation (6))} &> d-d^*-\mu(d-d^*) \\ &= (1-\mu)(d-d^*). \end{aligned} \tag{7}$$

From the chain growth, we know that $d-d^* \geq Q^P(S)$ and combining this with Equation (7) gives us

$$A^P(J) > (1-\mu)Q^P(S) = (1-\delta+3\varepsilon)Q^P(S), \tag{8}$$

which contradicts Lemma B.5. □

D PERSISTENCE AND LIVENESS OF OHIE

Lemma D.1 (Common prefix and chain quality for each individual chain). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment, each chain i ($0 \leq i \leq m-1$) satisfy the the common-prefix property with parameter $\ell_{\text{cp}} = \ell + 2\Delta$ and chain-quality property with parameter $\ell_{\text{cq}} = \ell + 2\Delta$ and $\mu = \delta - 3\varepsilon$.*

Proof. The proof directly follows Lemmas B.6,B.7, 6.13 and 6.14. □

We call all blocks on a chain except the blocks with timestamp in last $\ell + 2\Delta$ rounds as partially confirmed. Then by Lemma D.1, we know that the ordering of these partially confirmed blocks on their chain will no longer change in the future. Recall that OHIE generates a SCB in the following way. Consider any given honest node at any given time and its local view of all the m chains. Let y_i be the `next_rank` of the last partially-confirmed block on chain i in this view. Let `confirm_bar` $\leftarrow \min_{i=1}^k y_i$. Then all partially-confirmed blocks whose rank is smaller than `confirm_bar` are deemed as fully-confirmed, and included in SCB. Finally, all the fully-confirmed blocks will be ordered by increasing `rank` values, with tie-breaking favoring smaller chain ids. Next, we will prove that the SCB generated in this way satisfies persistence and liveness properties.

Theorem D.2 (Persistence and Liveness of OHIE). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment,* OHIE *satisfies persistence and liveness with parameter $u = 2\ell + 4\Delta$.*

Proof. We first prove the persistence property. Let L_1 and L_2 be the SCB held by two honest node u_1 and u_2 at round r_1 and r_2 respectively, with $r_2 > r_1 + \Delta$. Let x_1 and x_2 be the `confirm_bar` of L_1 and L_2, respectively. Then we will prove that L_1 is a prefix of L_2.

Let $F_1(i)$ be the sequence of partially-confirmed blocks on chain i in u_1's view at round r_1. Let $G_1(i)$ be the prefix of $F_1(i)$ such that $G_1(i)$ contains all blocks in $F_1(i)$ whose `rank` is smaller than x_1. Similarly define $F_2(i)$ and $G_2(i)$, where $G_2(i)$ contains those blocks in $F_2(i)$ whose `rank` is smaller than x_2. We first prove the following two claims:

- For all i, $0 \leq i \leq m-1$, $G_1(i)$ is a prefix of $G_2(i)$. By round r_2, node u_2 must have seen all blocks seen by u_1 at round r_1. By the common prefix property in Lemma D.1, all partially-confirmed blocks on u_1 at round r_1 must also be partially-confirmed on u_2 at round r_2, i.e., $F_1(i)$ is a prefix of $F_2(i)$. Then we have $x_1 \leq x_2$ and also $G_1(i)$ should be a prefix of $G_2(i)$.
- For all i, $0 \leq i \leq m-1$, and all block $B \in G_2(i) \setminus G_1(i)$, B's `rank` must be no smaller than x_1. We prove this claim via a contradiction and assume that B's `rank` is smaller than x_1. Together with the fact that B is in $G_2(i) \setminus G_1(i)$, we know that B is in $F_2(i)$ but not in $F_1(i)$. Let x_0 be the `rank` of the last block B_0 in $F_1(i)$. Since $F_1(i)$ is a prefix of $F_2(i)$, both B_0 and B must be in $F_2(i)$, and B must be a descendent of B_0 in $F_2(i)$. Since the blocks in $F_2(i)$ must have increasing `rank` values, we know that B_0's `rank` must be smaller than B's, hence we have $x_0 < x_1$. On the other hand, since B_0's `next_rank` must be greater than its `rank`, by the design of `confirm_bar`, we know that $x_1 < x_0$. This yields a contradiction.

Now we can use the above two claims to prove that L_1 is a prefix of L_2. Note that $L_1 = \cup_{0 \leq i \leq m-1} G_1(i)$ and $L_2 = \cup_{0 \leq i \leq m-1} G_2(i)$. Since $G_1(i)$ is a prefix of $G_2(i)$ for all i, we know that $L_1 \subseteq L_2$. For all blocks in L_1, the sequence L_1 orders them in exactly the same way as the sequence L_2. For all block $B \in L_2 \setminus L_1$, by the second claim above, we know that B's `rank` must be no smaller than x_1. Hence in the sequence L_2, all blocks in $L_2 \setminus L_1$ must be ordered after all the blocks in $L_2 \cap L_1$ (whose `rank` must be smaller than x_1). This completes our proof that L_1 is a prefix of L_2.

We next prove the liveness property. Suppose a transaction `tx` is received by all honest nodes before or at round r_0. Let L be the SCB of an honest node u at round $r \geq r_0 + 2\ell + 4\Delta$ and x be the `confirm_bar` of L. For $0 \leq i \leq m-1$, let $F(i)$ be the sequence of partially-confirmed blocks on chain i in u's view at round r and $G(i)$ be the prefix of $F(i)$ such that $G(i)$ contains all blocks in $F(i)$ whose `rank` is smaller than x. Without loss of generality, assume x is the `next_rank` of the last block in $F(0)$. Then we have $G(0) = F(0) \subset L$. By the chain quality property in Lemma D.1, we know that there exists at least one honest block B on chain 0 whose timestamp is in the interval $(r-2\ell-4\Delta, r-\ell-2\Delta)$. As B is an honest block mined after r_0, B or an ancestor of B must contain `tx`. Therefore L contains `tx` since B and all its ancestors are fully confirmed. □

E FAIRNESS OF FRUITCHAINS

For our analysis to go through, we choose the following protocol parameters for FruitChains:

$$\ell \triangleq \frac{4(1+3\varepsilon)}{\varepsilon^2 f[1-(1+\delta)\gamma^2 f]^{\Delta+1}} \max\{\Delta, \tau\}\gamma^4\lambda, \tag{9}$$

$$\Phi = 4(1+\delta)\gamma^2 f(\ell+3\Delta)/\varepsilon, \tag{10}$$

and

$$s = 2(1+\delta)\gamma^2\Phi/f. \tag{11}$$

Note that in Equation (9), we define a larger ℓ than in Equation (3) for Prism and OHIE. This is for the proof of fairness to work. Since Equation (10) satisfies Equation (4) and s is selected the same as in Prism and OHIE, all the pivot-chain properties stated in Appendix B still hold for FruitChains. Therefore, the persistence and liveness of variable difficulty FruitChains simply follow the persistence and liveness of Bitcoin, as the protocols are identical if we ignore the content of the blocks. In this section, we formally prove the fairness property guaranteed by FruitChains. To ease the analysis, we assume $t_r = (1-\delta)n_r$ for all rounds r, i.e., the number of adversarial queries in the worst case. We first give a definition of fairness.

Definition E.1. *We say that $\mathcal{H}$ is a ϕ-fraction honest subset if players in $\mathcal{H}$ (that may change over time) are honest and $n_r^{\mathcal{H}} = \lceil \phi(n_r+t_r) \rceil = \lceil \phi(2-\delta)n_r \rceil$ for round r, where $n_r^{\mathcal{H}}$ is number of players in $\mathcal{H}$ at round r.*

Definition E.2 (Fairness). *We say the* FruitChains *protocol has (approximate) $\sigma-$fairness if for any $0<\phi<\frac{1}{2-\delta}$ and any $\phi-$fraction honest subset $\mathcal{H}$, there exists $W_0 \in \mathbb{N}$ such that for any honest player holding chain C at round r and any interval $S_0 \in [0,r]$ with at least W_0 consecutive rounds, let $C(S_0)$ be the segment of C containing blocks with timestamps in S_0 and d be the total difficulty of all fruits included in $C(S_0)$, then it holds that the fruits included in $C(S_0)$ mined by $\mathcal{H}$ have total difficult at least $(1-\sigma)\phi d$.*

By the common prefix property proved in Lemma B.6, for a chain C held by an honest node at round r, the prefix $C^{\lceil \ell+2\Delta}$ are stabilized, so to mine a fruit/block at round r an honest miner will select the tip of $C^{\lceil \ell+2\Delta}$ as the fruit parent by our proposed variable difficulty FruitChains algorithm. And we set the recency parameter $R=3\ell+7\Delta$, i.e., a fruit B_f is recent w.r.t. a chain C at round r if the fruit parent of B_f is in C and has timestamp at least $r-3\ell-7\Delta$. With this selection of the recency parameter, we can prove the following key property of the FruitChains protocol: any fruit mined by an honest player will be incorporated into the stabilized chain (and thus never lost). We refer to this as the *Fruit Freshness Lemma*—fruits stay "fresh" sufficiently long to be incorporated.

Lemma E.3 (Fruit Freshness). *For a typical execution in a $(\gamma, 2(1+\delta)\gamma^2\Phi/f)$-respecting environment, if $R=3\ell+7\Delta$, then an honest fruit mined at round r will be included into the stabilized chain before round $r+r_{\text{wait}}$, where $r_{\text{wait}}=2\ell+5\Delta$.*

Proof. Suppose an honest fruit B_f is mined at round r_0 with block parent being the tip of C_0, then the fruit parent of B_f is the tip of $C_0^{\lceil \ell+2\Delta}$. Further, by the NoStaleChains property (defined in Appendix B), the fruit parent of B_f has timestamp $r_1 \geq r_0-2\ell-4\Delta$. By round $r_0+\Delta$, all honest nodes will receive B_f. Let $r_2=r_0+2\ell+5\Delta$ and C be any chain held by honest nodes at round r_2, then by the chain quality property in Lemma B.7, we know that there exists at least one honest block B on C whose timestamp r_3 is in the interval $(r_2-2\ell-4\Delta, r_2-\ell-2\Delta)$. We check that B_f is still recent at round r_3 as

$$r_3-r_1<(r_2-\ell-2\Delta)-(r_0-2\ell-4\Delta)=3\ell+7\Delta=R.$$

As B is an honest block mined after $r_0+\Delta$, B or an ancestor of B must include B_f. And since B is stabilized in C at round r_2, we have that $r_{\text{wait}}=r_2-r_0=2\ell+5\Delta$ □

Define the random variable D_r equal to the sum of the difficulties of all fruits computed by honest parties at round r. And for fixed ϕ and $\phi-$fraction honest subset $\mathcal{H}$, define the random variable $D_r^{\mathcal{H}}$ equal to the sum of the difficulties of all fruits computed by parties in $\mathcal{H}$ at round r. For a set of rounds S, we define $D(S)=\sum_{r\in S} D_r$, $D^{\mathcal{H}}(S)=\sum_{r\in S} D_r^{\mathcal{H}}$, and $n^{\mathcal{H}}(S)=\sum_{r\in S} n_r^{\mathcal{H}}$. For a set of J adversarial queries, define the random variable $A(J)$, as the sum of difficulties of all the fruits created during queries in J.

Next we define the following fruit-typical execution.

Definition E.4 (Fruit-typical Execution). *An execution E is fruit-typical if the followings hold*

(a) For any set S of at least ℓ consecutive good rounds,

$$D(S)<(1+\varepsilon)pn(S).$$

(b) For any set S of at least ℓ consecutive good rounds, let J be the set of adversarial queries in S, if we further know each query in J made at round r with target T satisfies $f/2\tau\gamma^3 \leq pn_rT \leq (1+\delta)\tau\gamma^3 f$, then

$$D(S)+A(J)<(1+\varepsilon)p(n(S)+|J|).$$

(c) For any set S of at least $\ell/\phi(2-\delta)$ consecutive good rounds,

$$D^{\mathcal{H}}(S)>(1-\varepsilon)pn^{\mathcal{H}}(S).$$

Theorem E.5. *For an execution $\mathcal{E}$ of L rounds, in a (γ,s)-respecting environment, the probability of the event "$\mathcal{E}$ not fruit-typical" is bounded by $O(L)e^{-\lambda}$.*

Proof. The proof for (a) is the same as in [13]. For (b), by the condition, for each query in S (made either by an honest node or the adversary) at round r with target T, we have $f/2\tau\gamma^3 \leq p(n_r+t_r)T \leq 2(1+\delta)\tau\gamma^3 f$. Therefore, the proof is similar to [13] (by setting ℓ to be slightly larger as we did in Eqn.(9)).

For (c), let the execution be partitioned into parts such that each part has at least $l/\phi(2-\delta)$ rounds and at most s rounds. We will prove that the statement fails with a probability less than $e^{-\lambda}$ for each part. Let J denote the queries made by $\mathcal{H}$ in rounds S. We have $|J|=n^{\mathcal{H}}(S)=n(S)\phi(2-\delta)$. For $k\in[|J|]$, let Z_i be the difficulty of any block obtained from query $j\in J$. Then

$$X_0=0$$
$$X_k=\sum_{i\in[k]} Z_i-\sum_{i\in[k]} \mathbb{E}[Z_i|\mathcal{E}_{i-1}]$$

is a martingale with respect to $\mathcal{E}_0,\ldots,\mathcal{E}_k$. We have

$$X_k-X_{k-1}=X_k-\mathbb{E}[X_k|\mathcal{E}_{k-1}]=Z_k-\mathbb{E}[Z_k|\mathcal{E}_{k-1}]$$
$$\leq \frac{1}{T_k^{min}}=\frac{pn_k}{pn_kT_k^{min}} \leq 2\gamma^3 pn(S)/f|S|:=b.$$

Similarly

$$V=\sum_k var[X_k-X_{k-1}|\mathcal{E}_{k-1}] \leq \sum_k \mathbb{E}[Z_k^2|\mathcal{E}_{k-1}]$$
$$=\sum_k pT_k\frac{1}{T_k^2} \leq 2\gamma^3p^2|J|n(S)/f|S|:=v.$$

Let the deviation $t=\varepsilon p|J|=\varepsilon\phi(2-\delta)pn(S)$, then we have $b=\frac{2\gamma^3 t}{\varepsilon\phi(2-\delta)f|S|}$ and $v=\frac{2\gamma^3 t^2}{\varepsilon^2\phi(2-\delta)f|S|}$. Using the minimum value of $|S|$ is $l/\phi(2-\delta)$ and applying Theorem C.1 to $-X_{|J|}$, we have

$$P[D(S)<(1-\varepsilon)pn^{\mathcal{H}}(S)] \leq \exp(-\frac{\varepsilon t}{2b(1+\varepsilon/3)}) \leq \exp(-\lambda).$$

This concludes the proof.

□

Theorem E.6 (Fairness). *For a typical and fruit-typical execution in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment, the* FruitChains *protocol with recency parameter $R=3\ell+7\Delta$ satisfies $\sigma-$fairness, where $\sigma=4\varepsilon$.*

Proof. Fixed ϕ, a ϕ–fraction honest subset $\mathcal{H}$, and an honest player holding chain C. Set $W_0 = \max\{s/2, \ell/\phi(2-\delta)\} + 3\ell + 7\Delta$. Let $S_0 = \{u : r_1 \le u \le r_2\}$ be a window of at least W_0 consecutive rounds. Let $C(S_0)$ be the segment of C containing blocks with timestamps in S_0, let $\mathcal{F}$ be all fruits included in $C(S_0)$, and d be the total difficulty of all fruits in $\mathcal{F}$. Then we have the following facts:

- **Fact 1.** For any $B_f \in \mathcal{F}$, B_f is mined after $r_1 - 4\ell - 9\Delta$. Indeed by recency condition, B_f's fruit/block parent has timestamp at least $r_1 - R$. By Accuracy property (defined in Appendix B), B_f's fruit/block parent is mined after $r_1 - R - (\ell + 2\Delta)$. So B_f must be mined after $r_1 - 4\ell - 9\Delta$.
- **Fact 2.** For any $B_f \in \mathcal{F}$, B_f is mined before $r_2 + \ell + 2\Delta$. Indeed the block including B_f has timestamp at most r_2, and by Accuracy property, the block including B_f is mined before $r_2 + \ell + 2\Delta$. So B_f must be mined before $r_2 + \ell + 2\Delta$.
- **Fact 3.** If a fruit B_f is mined by $\mathcal{H}$ after r_1 and before $r_2 - 3\ell - 7\Delta$, then $B_f \in \mathcal{F}$. Indeed, by NoStaleChain property, the last honest block in $C(S_0)$ has timestamp at least $r_2 - \ell - 2\Delta$. Hence by Lemma E.3, all honest fruit mined after r_1 and before $r_2 - \ell - 2\Delta - r_{\text{wait}}$ will be included into a block in $C(S_0)$.

Let $S_1 = \{u : r_1 - (4\ell + 9\Delta) \le u \le r_2 + (\ell + 2\Delta)\}$, $S_2 = \{u : r_1 \le u \le r_2 - (3\ell + 7\Delta)\}$, and J be the set of adversary queries associated with $\mathcal{F}$ in S_1. Then by Fact 1 and Fact 2, we have all fruits in $\mathcal{F}$ are mined in S_1; by Fact 3, we have all fruits mined by $\mathcal{H}$ in S_2 are in $\mathcal{F}$. Also note that for each query in J, to mine a recent fruit B_f with target T at round r, the timestamp of B_f's fruit parent B must be at least $r - R$. Let B_0 with target T_0 be the last honest ancestor of B, then B_0 will have timestamp $r_0 \ge r - R - (\ell + 2\Delta)$. By GoodRounds property (defined in Appendix B), we know $f/2\gamma^2 \le pn_{r_0}T_0 \le (1+\delta)\gamma^2 f$. By the bounded difficulty change rule, we have $T_0/\tau \le T \le \tau T_0$. And in a $(\gamma, 2(1+\delta)\gamma^2 m/f)$-respecting environment, $n_{r_0}/\gamma \le n_r \le \gamma n_{r_0}$. Therefore we have $f/2\tau\gamma^3 \le pn_r T \le (1+\delta)\tau\gamma^3 f$. Further note that, to prove σ-fairness, it suffices to show that

$$D^{\mathcal{H}}(S_2) \ge (1-\sigma)\phi(D(S_1) + A(J)).$$

Under a fruit-typical execution, we have

$$D^{\mathcal{H}}(S_2) > (1-\varepsilon)pn^{\mathcal{H}}(S_2) \ge (1-\varepsilon)\phi(2-\delta)pn(S_2),$$

and

$$D(S_1) + A(J) < (1+\varepsilon)p(n(S_1) + |J|) = (1+\varepsilon)(2-\delta)pn(S_1).$$

By our choice of W_0, we have $|S_2| \ge s/2$. Furthermore, we may assume $|S_2| \le s$. This is because we may partition S_2 in parts such that each part has size between $s/2$ and s, sum over all parts to obtain the desired bound. Then by Lemma C.2, we have

$$\begin{aligned} n(S_1) &\le (1 + \frac{\gamma|S_1 \setminus S_2|}{|S_2|})n(S_2) \\ &\le (1 + \frac{\gamma|8\ell + 18\Delta|}{s/2})n(S_2) \\ &< (1 + \frac{2\varepsilon}{\gamma^3})n(S_2) \\ &< (1 + 2\varepsilon)n(S_2). \end{aligned}$$

Finally, by setting $\sigma = 4\varepsilon$, we conclude the proof.

□

F PSEUDOCODE OF PRISM

Algorithm 1 Prism: Main

```
1:  procedure MAIN( )
2:      INITIALIZE()
3:      while True do
4:          $header, Ppf, Cpf$ = POWMINING()
5:          // Block contains header, parent, content and merkle proofs
6:          if header is a tx block then
7:              $block \leftarrow \langle header, txParent, txPool, Ppf, Cpf \rangle$
8:          else if header is a prop block then
9:              $block \leftarrow \langle header, prpParent, unRfTxBkPool, Ppf, Cpf \rangle$
10:         else if header is a block in voter blocktree $i$ then
11:             $block \leftarrow \langle header, vtParent[i], votesOnPrpBks[i], Ppf, Cpf \rangle$
12:         BROADCASTMESSAGE($block$)                                   ▷ Broadcast to peers
13: procedure INITIALIZE( )                                         ▷ All variables are global
14:     // Blockchain data structure $C = (prpTree, vtTree)$
15:     $prpTree \leftarrow genesisP$                                       ▷ Proposer Blocktree
16:     for $i \leftarrow 1$ $to$ $m$ do
17:         $vtTree[i] \leftarrow genesisM\_i$                                ▷ Voter $i$ blocktree
18:     // Parent blocks to mine on
19:     $prpParent \leftarrow genesisP$                              ▷ Proposer block to mine on
20:     for $i \leftarrow 1$ $to$ $m$ do
21:         $vtParent[i] \leftarrow genesisM\_i$                       ▷ Voter tree $i$ block to mine on
22:     // Block content
23:     $txPool \leftarrow \phi$                          ▷ Tx block content: Txs to add in tx bks
24:     $unRfTxBkPool \leftarrow \phi$                     ▷ Prop bk content1: Unreferred tx bks
25:     $unRfPrpBkPool \leftarrow \phi$                   ▷ Prop bk content2: Unreferred prp bks
26:     for $i \leftarrow 1$ $to$ $m$ do
27:         $votesOnPrpBks(i) \leftarrow \phi$                           ▷ Voter tree $i$ bk content
```

Algorithm 2 Prism: Mining

```
1: procedure PowMining( )
2:     while True do
3:         $txParent \leftarrow prpParent$
4:         // Assign content for all block types/trees
5:         for $i \leftarrow 1\ to\ m$ do $vtContent[i] \leftarrow votesOnPrpBks$[i]
6:         $txContent \leftarrow txPool$
7:         $prContent \leftarrow (unRfTxBkPool, unRfPrpBkPool)$
8:         // Define parents and content Merkle trees
9:         $parentMT \leftarrow$MerklTree($vtParent, txParent, prpParent$)
10:        $contentMT \leftarrow$MerklTree($vtContent, txContent, prContent$)
11:        nonce $\leftarrow$ RandomString($1^\kappa$)
12:        // Header is similar to Bitcoin
13:        header $\leftarrow \langle$ $parentMT$.root, $contentMT$.root, nonce $\rangle$
14:        if chainLength($prpParent$) % e == 0 then
15:            $f_p^{new} \leftarrow$ RecalculateTarget($f_p$)
16:            $f_v \leftarrow (f_v * f_p^{new}/f_p)$
17:            $f_t \leftarrow (f_t * f_p^{new}/f_p)$
18:            $f_p \leftarrow f_p^{new}$
19:        // Sortition into different block types/trees
20:        if Hash(header) $\leq mf_v$ then                          ▷ Voter block mined
21:            $i \leftarrow \lfloor$Hash(header)/$f_v\rfloor$ and break              ▷ on tree $i$
22:        else if $mf_v <$Hash(header) $\leq mf_v + f_t$ then
23:            $i \leftarrow m+1$ and break                            ▷ Tx block mined
24:        else if $mf_v + f_t <$ Hash(header) $\leq mf_v + f_t + f_p$ then
25:            $i \leftarrow m+2$ and break                            ▷ Prop block mined
    // Return header along with Merkle proofs
26:    return $\langle header, parentMT$.proof($i$), $contentMT$.proof($i$)$\rangle$
```

Algorithm 3 Prism: Block and Tx handling

```
1: procedure ReceiveBlock(B)                                   ▷ Get block from peers
2:     if B is a valid transaction block then
3:         $txPool$.removeTxFrom(B)
4:         $unRfTxBkPool$.append(B)
5:     else if B is a valid block on $i^{th}$ voter tree and ValidVote(B,i) then
6:         $vtTree[i]$.append(B) and $vtTree[i]$.append(B.ancestors())
7:         // A vote is a range of difficulty along with the the corresponding proposer block
8:         if B.chaindiff > $vtParent[i]$.chaindiff then
9:             $vtParent[i] \leftarrow$ B and $votesOnPrpBks(i)$.update(B)
10:    else if B is a valid prop block then
11:        if B.diff > $prpParent$.diff then
12:            $prpParent \leftarrow$ B
13:            for $i \leftarrow 1\ to\ m$ do                       ▷ Add vote on level $\ell$ on all $m$ trees
14:                $votesOnPrpBks$(i)[B.$level$] $\leftarrow$ B
15:        else if B.level > $prpParent$.level+1 then
16:            // Miner doesnt have block at level $prpParent$.level+1
17:            RequestNetwork(B.parent)
18:        $prpTree$[B.level].append(B),  $unRfPrpBkPool$.append(B)
19:        $unRfTxBkPool$.removeTxBkRefsFrom(B)
20:        $unRfPrpBkPool$.removePrpBkRefsFrom(B)
21: procedure ReceiveTx(tx)
22:     if tx has valid signature then $txPool$.append(B)
```

Algorithm 4 Prism: Vote validation

```
procedure ValidVote(B,i )                                                    ▷ validate a vote
    // voter block can't vote for difficulty grater than its proposer parent
    if B.vtContent[i].latestBlock.chaindiff > B.prpParent.chaindiff then
        return False
    if B.vtContent[i] has discontinuous votes then
        return False
    if B.vtContent[i].earliestBlock.parent.chaindiff > B.vtParent[i].chaindiff then
        return False
    // include the check where the difficulty ranges of the votes should end at proposal blocks
    return True
```

Algorithm 5 Prism: Tx confirmation

```
procedure IsTxConfirmed($tx$)
    $\Pi \leftarrow \phi$                                                     ▷ Array of set of proposer blocks
    for $\ell \leftarrow 1$ to $prpTree$.maxLevel do
        $votesNdepth \leftarrow \phi$
        for $i$ in 1 to $m$ do
            $votesNdepth[i] \leftarrow$ GetVoteNDepth($i, \ell$)
        if IsPropSetConfirmed($votesNdepth$) then
            $\Pi[\ell] \leftarrow$ GetProposerSet($votesNdepth$)
        else break
    // Ledger list decoding: Check if tx is confirmed in all ledgers
    $prpBksSeqs \leftarrow \Pi[1] \times \Pi[2] \times \cdots \times \Pi[\ell]$                              ▷ outer product
    for $prpBks$ in $prpBksSeqs$ do
        $ledger$ = BuildLedger($prpBks$)
        if $tx$ is not confirmed in $ledger$ then return False
    return True                                  ▷ Return true if tx is confirmed in all ledgers
// Return the vote of voter blocktree $i$ at level $\ell$ and depth of the vote
procedure GetVoteNDepth($i, d$)
    $voterMC \leftarrow vtTree[i].HeaviestChain()$
    for $voterBk$ in $voterMC$ do
        for $vote$ in $voterBk$.votes do
            if $d$ in $vote.range$ then
                // Depth is the difficulty of children bks of voter bk on main chain
                return ($vote.prpBk$, $voterBk$.depth)
procedure BuildLedger($propBlocks$)                                      ▷ Input: list of prop blocks
    $ledger \leftarrow []$                                                  ▷ List of valid transactions
    for $prpBk$ in $propBlocks$ do
        $refPrpBks \leftarrow prpBk$.getReferredPrpBks()
        // Get all directly and indirectly referred transaction blocks.
        $txBks \leftarrow$ GetOrderedTxBks($prpBk, refPrpBks$)
        for $txBk$ in $txBks$ do
            $txs \leftarrow txBk$.getTxs()                                       ▷ Txs are ordered in $txBk$
            for $tx$ in $txs$ do
                // Check for double spends and duplicate txs
                if $tx$ is valid w.r.t to $ledger$ then $ledger$.append($tx$)
    return $ledger$
// Return ordered list of confirmed transactions
procedure GetOrderedConfirmedTxs()
    $\mathsf{L} \leftarrow \phi$                                                  ▷ Ordered list of leader blocks
    for $prpBk$ in $propBlocks$ do
        $g(p) = inf_d(d : \text{GetLeader}(d) = p)$
    $L \leftarrow sort(p, key = g(p))$
    return BuildLedger(L)
```

G PSEUDOCODE OF OHIE

Algorithm 6 OHIE: Mining

```
1:  $T \leftarrow$ INITIAL DIFFICULTY;
2:  $V_i \leftarrow$ {(genesis block of chain $i$, the attachment for genesis block of chain $i$)}, for $0 \le i \le m-1$;
3:  $M \leftarrow$ Merkle tree of the hashes of the $m$ genesis blocks;
4:  trailing $\leftarrow$ hash of genesis block of chain 0;
5:
6:  procedure OHIE( )
7:      repeat forever {
8:          ReceiveState();
9:          Mining();
10:         SendState();
11:     }
12:
13: procedure MINING
14:     $B$.transactions $\leftarrow$ get_transactions();
15:     $B$.root $\leftarrow$ root of Merkle tree $M$;
16:     $B$.trailing $\leftarrow$ trailing;
17:     $B$.nonce $\leftarrow$ new_nonce();
18:     $\widehat{B}$.hash $\leftarrow$ $hash(B)$;
19:     if ($\widehat{B}$.hash $< T$) {
20:         $i \leftarrow$ last $\log_2 m$ bits of $\widehat{B}$.hash;
21:         $\widehat{B}$.leaf $\leftarrow$ leaf $i$ of $M$;
22:         $\widehat{B}$.leaf_proof $\leftarrow$ $M$.MerkleProof($i$);
23:         $\widehat{B}$.chain0_parent $\leftarrow$ leaf 0 of $M$;
24:         $\widehat{B}$.parent_proof $\leftarrow$ $M$.MerkleProof(0);
25:         ProcessBlock($B$, $\widehat{B}$);
26:     }
27:
28: procedure SENDSTATE
29:     send $V_i$ $(0 \le i \le m-1)$ to other nodes;        ▷ In implementation, only need to send those blocks not sent before.
30: procedure RECEIVESTATE
31:     foreach $(B, \widehat{B}) \in$ received state do
32:         ProcessBlock($B$, $\widehat{B}$);
33:     // a block $B_1$ should be processed before $B_2$, if $\widehat{B_2}$.leaf, $\widehat{B_2}$.chain0_parent or $\widehat{B_2}$.trailing points to $B_1$.
```

Algorithm 7 OHIE: ProcessBlock

```
procedure ProcessBlock($B$, $\widehat{B}$)
    // do some verifications
    $i \leftarrow$ last $\log_2 m$ bits of $\widehat{B}$.hash;
    verify that $\widehat{B}$.chain0_parent is leaf 0 in the Merkle tree, based on $B$.root and $\widehat{B}$.parent_proof;
    verify that $\widehat{B}$.chain0_parent $= \widehat{D}$.hash for some block $(D, \widehat{D}) \in V_0$;
    $T \leftarrow$ mining difficulty of $D$'s next block;
    verify that $\widehat{B}$.hash $< T$;
    verify that $hash(B) = \widehat{B}$.hash;
    verify that $\widehat{B}$.leaf is leaf $i$ in the Merkle tree, based on $B$.root and $\widehat{B}$.leaf_proof;
    verify that $\widehat{B}$.leaf $= \widehat{A}$.hash for some block $(A, \widehat{A}) \in V_i$;
    verify that $B$.trailing $= \widehat{C}$.hash for some block $(C, \widehat{C}) \in \cup_{j=0}^{m-1} V_j$;
    $\widehat{B_0} \leftarrow \widehat{B}$'s parent block on chain $i$;
    $\widehat{D_0} \leftarrow \widehat{B_0}$.chain0_parent;
    verify that chaindiff($\widehat{D}$) $\geq$ chaindiff($\widehat{D_0}$);
    if (any of the above 8 verifications fail) then return;

    // compute rank and next_rank values
    $\widehat{B}$.rank $\leftarrow \widehat{A}$.next_rank;
    $\widehat{B}$.next_rank $\leftarrow \widehat{C}$.next_rank;
    if ($\widehat{B}$.next_rank $\leq \widehat{B}$.rank) then $\widehat{B}$.next_rank $\leftarrow \widehat{B}$.rank + diff($\widehat{B}$);

    // update local data structures
    $V_i \leftarrow V_i \cup \{(B, \widehat{B})\}$;
    update trailing;
    update Merkle tree $M$;
    if (chainLength(chain 0) % e == 0) then $T \leftarrow$ RecalculateTarget($T$)    ▷ update mining difficulty
```

Algorithm 8 OHIE: GenerateSCB

```
procedure OutputSCB
    // determine partially-confirmed blocks and confirm_bar
    for ($i = 0$; $i < m$; $i$++) {
        $W_i \leftarrow$ get_heaviest_path($V_i$);
        partial$_i$ $\leftarrow$ blocks in $W_i$ that are partially-confirmed;
        $(B_i, \widehat{B_i}) \leftarrow$ the last block in partial$_i$;
        $y_i \leftarrow \widehat{B_i}$.next_rank;
    }
    all_partial $\leftarrow \cup_{i=0}^{m-1}$ partial$_i$;
    confirm_bar $\leftarrow \min_{i=0}^{m-1} y_i$;

    // determine fully-confirmed blocks and SCB
    $L \leftarrow \emptyset$;
    foreach $(B, \widehat{B}) \in$ all_partial {
        if ($\widehat{B}$.rank $<$ confirm_bar) then $L \leftarrow L \cup \{(B, \widehat{B})\}$;
    }
    sort blocks in $L$ by rank, tie-breaking favoring smaller chain id;
    return $L$;
```