Adaptively Secure Fast Settlement with Dynamic Participation and Self-Healing

Christian Badertscher[1], Sandro Coretti[2], Peter Gaži[2], Aggelos Kiayias[3], and Alexander Russell[4]

[1] Zurich University of Applied Sciences & Input Output
christian.badertscher@{zhaw.ch,iohk.io}
[2] Input Output
firstname.lastname@iohk.io
[3] The University of Edinburgh & Input Output aggelos.kiayias@{ed.ac.uk,iohk.io}
[4] University of Connecticut & Input Output
acr@cse.uconn.edu

Abstract. An important property of blockchain systems is the time it takes for a block to enter the irreversible part of the chain, which is typically captured by the notions of finality or settlement. To date, only Nakamoto-style protocols are known to simultaneously offer (1) resilience against adaptive adversaries controlling up to less than half of the underlying resource (such as stake or computational power), (2) dynamic participation (where parties join and leave at will without announcement), and (3) self-healing, i.e., being able to recover from temporary periods of adversarial majority without external intervention. However, these protocols offer only probabilistic settlement that requires wait time linear in a security parameter. The following open question thus arises: can a protocol be designed that maintains these properties—50%-resilience to adaptive attacks, dynamic participation, and self-healing—while offering faster settlement?

This work answers the above question by proposing a novel Nakamoto-style proof-of-stake (PoS) protocol—Ouroboros Peras—that builds on Ouroboros, the proof-of-stake algorithm behind the Cardano Blockchain. Our protocol augments the chain-selection rule to incorporate a weight-based criterion and features a mechanism for certifying blocks, which increases their weight. Under favorable conditions, Peras continuously certifies new blocks in rapid succession, thereby significantly accelerating settlement time. If these conditions are not met, Peras' settlement guarantees gracefully fall back to those offered by Ouroboros. A unique feature of our approach is that, compared to prior approaches based on finality gadgets, our settlement acceleration mechanism retains the self-healing and dynamic availability properties of Nakamoto-style protocols.

1 Introduction

The development of blockchain protocols since the inception of Bitcoin [30] in 2008 has produced a wide range of designs that differ in their fundamental security assumptions, their resilience under low participation or temporary adversarial dominance, and their operational performance metrics, such as latency and throughput. Consequently, selecting a particular design involves navigating trade-offs across a complex set of interacting dimensions.

Longest-chain consensus, also referred to as Nakamoto-style consensus in honor of the pseudonymous inventor of Bitcoin, is a permissionless design in which participants engage in a lottery—either proof-of-work or proof-of-stake—to append the next block to the longest chain. The longest chain exhibits a common-prefix property: while the most recent blocks may be transient, the probability that a block is reverted decreases as additional blocks are built on top of it, i.e., as it receives more confirmations. The probabilistic and gradual notion of finality introduced by Bitcoin was novel and contrasted with traditional approaches, particularly in the Byzantine fault-tolerance literature, where finality is typically treated as an absolute guarantee. From an operational perspective, block production in Bitcoin, as well as in its proof-of-stake counterpart Ouroboros, cannot proceed too quickly relative to network delay, since security relies on lottery successes being sufficiently "rare" events on the timescale of block propagation. In Cardano, the expected block time is approximately 20 seconds, whereas in Bitcoin it is 10 minutes, directly affecting both latency and throughput.

A unique feature of longest-chain consensus, established formally through a sequence of works [1,2], is that it simultaneously achieves three important robustness properties: *consistency and liveness under dynamic availability*, *self-healing*, and *the ability to bootstrap from Genesis*. More precisely, longest-chain consensus remains both live and safe under dynamic participation, meaning that blockchain growth (and consequently confirmation times) adapts to the level of active participation. The common-prefix property holds as long as a majority of the resources controlled by active participants—measured as hash power in Bitcoin or stake in Cardano—remains honest.

Moreover, even if an attacker temporarily controls a majority of the active resources, for example due to reduced honest participation, the protocol can recover and return to normal operation once this period ends. This property, while seemingly theoretical at first sight, has proven to be quite important in practice, witnessed by a recent incident on the Cardano blockchain [27]. In this particular case, the protocol was able to recover from an adverse condition without external intervention, which implied that higher-level applications, which did experience issues during the spike, could automatically heal as a consequence of the healed chain—something that is generally not possible with external intervention (such as blacklisting bad blocks or chains), which result in updates of third-party applications to remain consistent with the blockchain.

While guarantees against adversaries of a priori unbounded duration hold only for longest-chain proof-of-stake systems with static stake, the self-healing property for dynamic-stake (multi-epoch) proof-of-stake protocols remains valid provided the attack duration is confined to a single epoch. Put differently, for a desired level of robustness against majority attacks, the protocol can be parameterized to ensure self-healing. Finally, longest-chain consensus protocols provide a clear rule set enabling newly joining participants to identify the "best chain" using only knowledge of the genesis block, thereby eliminating the need for checkpointing recent blocks.

While its security guarantees are unparalleled, longest-chain consensus can be less operationally efficient, particularly with respect to finality. Although longest-chain consensus

offers user-defined finality, achieving a very high level of assurance that a payment will not be reverted may require waiting several hours because a large number of confirmations must accumulate. This motivates the central question of this work:

Is it possible to accelerate confirmation times in longest-chain consensus while retaining all of the robustness properties described above?

The importance of this question arises from the fact that these properties appear contradictory at first sight. Traditional Byzantine fault-tolerant (BFT) protocols [6,14], as well as blockchain protocols derived from BFT techniques such as Jolteon [25] and Algorand [7], can in principle finalize blocks at network speed by reaching a quorum. However, to date there exists no standalone iterated-BFT-based blockchain protocol that can simultaneously adjust quorum size to the actual level of participation—thereby enabling progress under dynamic availability, as explored in recent work by Malkhi et al. [29]—and recover from situations in which honest active participants temporarily constitute a minority. The underlying reason is fundamental: achieving both properties would require the protocol to be capable of resolving two conflicting finalizations, implying that the system could not have been in a univalent state with certainty in the first place. In this respect, Nakamoto-style consensus, with its probabilistic guarantees, is better suited to achieving these properties simultaneously. We also note, and discuss in detail in Section 1.2, that even approaches in which a BFT protocol is deployed as an overlay on top of a Nakamoto-based blockchain, such as Afgjort [13], encounter the same limitation whenever finality established by the overlay protocol takes precedence over the longest-chain rule.

1.1 Our Results

We answer the above question in the affirmative and introduce *Ouroboros Peras*, a longest-chain proof-of-stake protocol. In comparison to its predecessors, the protocol introduces a parallel process that selectively boosts certain blocks on the longest chain, thereby giving them an advantage in the Nakamoto race. To enable this, blocks are no longer treated as having equal weight (a concept already implicit in Bitcoin); instead, we adopt a weight-based view of the chain. Ordinary blocks have unit weight, whereas boosted blocks receive an additional weight of B, where B is a configurable parameter. In contrast to prior designs that introduce "finality overlays" on top of longest-chain consensus protocols, our approach preserves the core Nakamoto logic: participants always follow the heaviest chain, and a block is considered settled once it is buried under sufficient accumulated weight. This allows us not only to re-establish the standard common-prefix guarantee under an honest majority in the dynamic participation model, but also to support a weight-based bootstrapping procedure for newly joining participants. Moreover, the protocol retains a self-healing property, enabling it to recover from dishonest-majority attacks of bounded strength through appropriate parameter configuration.

We provide a rigorous formal security analysis of the new protocol, which turns out to be technically demanding. Although our boosting gadget can be understood as a simple voting mechanism for a preferred block—reminiscent of earlier approaches surveyed in Section 1.2—these additional votes interact with the "plain" Nakamoto execution in subtle and non-trivial ways. While blocks with strong support are indeed boosted, our formal treatment must ensure that votes cannot be abused either to preferentially amplify adversarial chains or to cause honest participants to expend slots on chains that are inferior to the heaviest chain. For this reason, the boosting gadget enters a "cooldown" period after failed boost attempts

are detected. This mechanism is crucial for establishing both consistency and liveness of the combined protocol, since unsuccessful boosting attempts could otherwise be exploited to the adversary's advantage.

1.2 Related Work

In the academic literature, the task of boosting the finality guarantees of a Nakamoto-style blockchain has been considered in a sequence of papers that date back to Thunderella [33], Polkadot's Grandpa protocol [35], Casper [5], as well as Afgjort [13]. The overarching idea of this sequence of works is that in a typical, real-life execution, the *actual* common prefix of the underlying Nakamoto-chain is in fact much longer than what the security analysis indicates—in which case the finality gadget's task is to identify this common prefix and to finalize it. We detail the comparison to each class of protocols below.

Another related line of work is the Ebb-and-Flow protocols [31], which, however, do not aim at finalizing the Nakamoto-chain, but at combining different security guarantees in one protocol. An Ebb-and-Flow protocol thus provides the user with an adaptive choice of what level of finality they seek in an application, such as choosing either a consistency-first approach (in which users essentially never observe reverted transactions, even if the network degrades) or a liveness-favoring rule with a probabilistic Nakamoto-style guarantee, where transactions could be reverted in case of network disruptions. In our comparison below, we focus mainly on finality gadgets that aim to reduce finalization times for Nakamoto blockchains, but we also include relevant work on hybrid consensus and checkpointing gadgets.

Finally, a third approach is based on a set of *parallel chains* that proceed in tandem which can be used to infer a combined ledger that can offer faster settlement guarantees than what each one of them individually could [15,3,16].

Afgjort finality layer. Afgjort's [13] approach is to let a committee-based BA protocol try to reach agreement on which block in the common-prefix has high support. If successful, the block is declared as final and the chain-selection rule of the Nakamoto-blockchain is changed to comply with that decision, i.e., any blockchain not containing these finalized blocks gets invalidated. This design choice has two implications that we rectify with our solution: first, changing Nakamoto chain-selection to always comply with finalized blocks implies that any equivocation from the finalization committee—due to adversarial dominance, which is not an extremely unlikely event for those committee sizes occurring in practice—leads to a permanent split of parties, rendering the combined protocol into a protocol that only tolerates a corruption threshold of $1/3$, and new features would be needed to return to normalcy after the assumption is violated. The second implication is lack of resilience against dynamic participation: if the finality gadget is stalled for an extended period of time due to committee members being temporarily unavailable, the underlying Nakamoto-chain would still grow substantially. However, if the committee members come back online, there is the risk that they finalize an old block that is actually not on the common prefix of parties running the Nakamoto-chain. Since the design favors the finalization block, a large portion of the chain will be reverted even in a setting without a dishonest (relative) majority, which appears undesirable.

In contrast to Afgjort, our solution does not overrule the Nakamoto-chain. Its approach is to selectively boost the weight of blocks—giving them an advantage to win—but the final call is done by the Nakamoto-blockchain. We retain the honest majority threshold from the

underlying blockchain, and further inherit the self-healing guarantees [2]. Furthermore, if our voting committee gets stalled, our gadget gets temporarily turned off for sufficiently long, such that a very late-finish of a previous voting attempt would not lead to a reorganization of the chain, since the weight they contribute is too little to catch up with the longest chain at that time.

Grandpa finalization gadget. Grandpa is an elegant idea for finalizing blocks of an underlying longest-chain protocol. The idea is that parties in the finalization committee vote for the tip of their longest chain. Votes apply "recursively" and the collection of all votes can be used as an indicator on which blocks in the blocktree are supported by what fraction of the committee. If a block crosses a threshold, it will be marked as final and the longest-chain protocol complies with that decision in its chain selection rule as above. Note that voting always succeeds under full participation due to the common-prefix guarantee of the underlying blockchain. In comparison to Grandpa, our solution removes two specific concerns: first, as in the case of Afgjort, the protocol overwrites the longest-chain chain selection rule, which downgrades the protocol to 1/3 security, the threshold of the finality layer, and once violated, the blockchain would fork permanently without additional intervention. Second, Grandpa assumes full participation, and it appears that under a temporary drop in participation, the protocol risks stalling. Furthermore, in such a situation, it is up to the adversary to release votes to resolve the situation. However, at this point, the adversary has the power to tip the scales adaptively to decide whether an "old block" is finalized (which always exists) or a "recent block". This is a danger to chain quality, as the adversary is given enough slack to adaptively discourage honestly generated blocks and wait to finalize adversarial blocks in a next round.

In contrast to that, our solution does not have the above disadvantages. In addition to the above comparison with Afgjort, our approach does not admit such a slack to the adversary since votes are bound to a block. Furthermore, as in Grandpa, in the optimistic case, the finalization rounds are at least as fast as chain-growth, and in case of failed vote, we retain at least the chain quality of the Nakamoto-Blockchain. More broadly speaking, our overall protocol retains security under fluctuating and low participation.

Thunderella and hybrid consensus. Hybrid consensus [32] and the related blockchain protocol Thunderella are closely related to the concept of finality gadgets. The main conceptual difference between finality gadgets and hybrid consensus is that a finality gadget tries to equip the blockchain protocol itself with a fast (and optimistic) path to settlement, but the blockchain remains the only ledger, without the need to sanitize the potentially different views of the fast and the slow settlement methods [31]. However, when it comes to concrete techniques used, the two problems also share similarities and a finality layer can typically be understood as providing some sort of hybrid consensus, but without producing its own blocks and using existing blocks from the Nakamoto-chain. If finality is reached for a block on the Nakamoto-chain, this block can further be used to bootstrap a new committee, and no dependency on the worst-case settlement time of the blockchain arises from committee selection [13].

When it comes to the security model, our solution and Thunderella share a few similarities and a couple of differences. First and foremost, our design does not use a public leader schedule and is designed with full adaptive security in mind. While Thunderella could switch to an adaptive security model, this would include rotating leaders and necessarily trigger a cool-down when a malicious leader is speaking, with a delay proportional to the actual

corruption threshold. In contrast, our solution comes with a set of parameters that allows one to make an informed tradeoff between the probability of actually cooling down (when the system is up and running against an adversary of a certain strength) and the settlement speed perceived by a transaction on the optimistic path.

Casper, Ebb-and-Flow, Goldfish. Casper [5] is presumably the first finality gadget proposed tailored to the Ethereum blockchain and based on a public committee that locks coins in order to be eligible to assist in finalizing (non-recent) blocks of an underlay chain, and in this regard belongs to the category of checkpointing layers, more formally known these days as Ebb-and-Flow protocols [31]. Such a layered approach ensures that the blockchain protocol maintains two ledgers: the dynamically available ledger (LMD GHOST in case of Ethereum, or some other longest-chain protocol [31,8]), and the accountable ledger which is a finalized prefix. To obtain a provably secure combination for the two-layer approach taken by Ethereum, Goldfish has been presented as a provably secure underlay following the LMD-GHOST approach. What makes Goldfish interesting in our comparison is that it does not only offer what is called standard k-deep confirmations of a prefix (against an adaptive adversary), but it contains an optimistic fast path to confirmations under good participation, honesty, and network conditions. While, of course, Goldfish is a consensus protocol on its own, and not an overlay over a Nakamoto-blockchain, it is interesting to observe some of the features that do make its approach not directly applicable to our setting due to our goal of achieving self-healing and bootstrapping from Genesis. First, the protocol, and extensions of it [9], are based on a short-term view of votes (i.e., votes expire) in such a way that if the set of unexpired votes is not dominated by a (relative) honest majority, this would lead to a protocol failure. Second, the Goldfish design does not specify a bootstrapping mechanism based solely on the Genesis block (under stake shift) and it remains open how to achieve it under the security model put forth in those papers.

Tezos. Due to the techniques used in our work, it is worth mentioning that the act of voting for a proposed block can also be used in other contexts for related but essentially different reasons. The Tezos blockchain [5] uses votes as indications of support, which allow a next block producer to publish a block quicker, thereby reducing blocktime. We note in passing that other uses of block votes have been proposed in Tezos for scalability reasons, which culminated in the above-mentioned approach.

Parallel chains. In the parallel-chains approach [15,3,16] a large number of longest-chain protocols proceed in tandem via a technique that allows mining simultaneously in all of them, see 2-for-1 PoWs in [19]. The key concept is that if a transaction is submitted to all chains simultaneously, it is possible to observe an event that in constant time has overwhelming probability of happening and ensures the transaction can be serialized in a combined ledger so that it is final. Early works on the topic suffered from the deficiency that double spending (or, in general, conflicting) transactions could slow down the settlement process to the same speed as the underlying longest chain protocol. This question was resolved in later work in parallel chain protocol design, first in [17] and then [18] in the dynamic availability setting. The key idea in these works was to interpret individual chains as virtual participants in a "phase-king" type of protocol (dubbed "chain-king") that may be corrupted with a certain probability of failure and use one fixed chain as the fall-back "king" when honest parties detect a possible discrepancy. In comparison to our solution, this line of work does not

[5] https://tezos.com/

consider the self-healing property; furthermore, early parallel chain works do not offer fast settlement for conflicting transactions; more recent instantiations do but with a penalty of increased message complexity (due to the large number of chains running in parallel), leaving open the real-world practicality of these designs.

2 Notation and Model

Basic notation. For a string $w = w_1 \dots w_N \in \Sigma^N$ we denote by $w_{k\rceil} := w_1 \dots w_k$ its prefix of length k. We denote by ε the empty string.

Time. For convenience, we model time as progressing in discrete steps called *slots* indexed with natural numbers $\mathbb{N} = \{1, 2, \dots\}$.

Network. We adopt a standard network model commonly used in the analysis of distributed algorithms, characterized by a single parameter Δ. The model guarantees that whenever an honest node sends or receives a message m in slot s, every other honest node receives m no later than the beginning of slot $s + (\Delta + 1)$. For convenience, and in line with prior work, we assume that parties diffuse entire chains in this manner. In practice, an implementation would reduce communication overhead by transmitting only the blocks necessary for a peer to catch up. Note that whenever a message m is injected for diffusion, the adversary is activated and may deliver the message to other honest parties at any time, subject only to the Δ delay bound. (In contrast, adversarial blocks may be delivered with arbitrary delays.) Consequently, the model provides no guarantee of a common serialization of messages.

Protocol executions. Given a consensus protocol Π, an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$, we define an execution as the random variable that contains the sequence of states of all activated parties in every slot. The environment $\mathcal{Z}$ activates a party by providing some transactions to be processed as well as the index of the current slot; it also activates the adversary $\mathcal{A}$ which is assumed to be activated always at the beginning of a slot before other parties are activated as well as after all parties have been activated in a slot. Note that the environment advances slots in a monotonically increasing manner, i.e., once a party is activated at slot i, all other parties will also be activated in slot i or larger. Parties use the network and the clock functionalities and terminate their activation by performing any updates dictated by Π in their internal stateThe adversary, besides interfering with message delivery, can also corrupt parties by issuing a special corruption instruction at any time during the execution. Once a corruption takes place, the adversary is activated instead of the party and has access to the internal state of the party. Given any execution it is possible to map it to a string which determines the number of honest and adversarial parties that have been activated.

Consistency and Liveness. In each slot, every party outputs (C', C), where C' is the part of C that it considers "settled." These chains must satisfy *consistency* and *liveness*. These are defined via the following adverse events:

- Consistency failure is a predicate $\mathsf{ConsFail}(\ell_1, \ell_2)$ defined for two slots $\ell_1 \leq \ell_2$ that is true if and only if there is an honest party in slot ℓ_1 that outputs a chain C_1' and (a) the same honest party in slot ℓ_2 outputs a chain C_2' such that C_1' is not a prefix of C_2'; or (b) the same or a different honest party outputs a chain C_2' in slot ℓ_2 such that neither C_1' nor C_2' is a prefix of the other.

– Liveness failure with parameter u is a predicate $\mathsf{LiveFail}_u(\ell)$ for a slot ℓ that is true if and only if a transaction tx was given for inclusion by the environment to all honest parties for u slots starting at slot ℓ but some honest party in slot $\ell + u$ outputs C that does not contain tx.

Definition 1. *A blockchain protocol Π satisfies* consistency and liveness *with error ϵ_{con} and ϵ_{liv} respectively if and only if for any L polynomial in security parameter κ, the predicates $\mathsf{ConsFail}(\ell_1, \ell_2)$ and $\mathsf{LiveFail}_u(\ell)$ for any $\ell, \ell_1, \ell_2 \leq L$ hold with probability ϵ_{con} and ϵ_{liv} respectively.*

Assumptions underlying security. Independently of any specific protocol, it is impossible to guarantee consistency or liveness unless restrictions are placed on how the adversary may influence the execution. A well-known result for Bitcoin, for example, is that security holds only when the majority of the hashing power invested in any given round is controlled by honest participants. For proof-of-stake (PoS) protocols, which we review in the following section, the relevant resource is stake.

For any round (j), we denote by $n_{\mathsf{h}}^{(j)}$ (respectively, $n_{\mathsf{a}}^{(j)}$) the number of activated honest (respectively, adversarial) resource units (such as hash queries in PoW or coins in PoS) during round (j). In the case of $n_{\mathsf{h}}^{(j)}$, we count only honest parties that are sufficiently synchronized with the protocol state to contribute to its security (cf. [1]). The "honest (super-)majority of resources" assumption, which underlies the security of all ledger consensus protocols, can now be stated as follows. For constants $\epsilon, \theta \in (0, 1]$, we say that an adversary satisfies the honest-majority assumption with parameters θ and ϵ over rounds $(i), \ldots, (j)$ if, for every round $r \in \{(i), \ldots, (j)\}$,

$$n_{\mathsf{a}}^{(r)} \leq (1 - \epsilon) \cdot \theta \cdot n_{\mathsf{h}}^{(r)} \tag{1}$$

holds. Typically, one fixes an arbitrary constant $\epsilon > 0$ together with a suitable threshold parameter $\theta \leq 1$ that captures the required level of (super-)majority.

In a similar manner, one may impose a minimum participation condition by requiring that $n_{\mathsf{h}}^{(j)} + n_{\mathsf{a}}^{(j)} > n_0$, where n_0 is a lower bound on the number of actively invested resource units at any given time (e.g., a lower bound on the number of hash queries or on the amount of active stake). Using this terminology, existing security guarantees for ledger protocols can be restated as asserting that the probability of a consistency or liveness violation is negligible (as a function of the relevant security parameters) for any adversary that satisfies the above honest-majority and participation constraints.

Self-healing. Given a protocol execution, an adversarial "spike" occurs in a slot when Equation (1) is violated for certain period of time. We note that in this paper, we consider this situation as arising due to fluctuating participation, as a consequence of very low honest participation resulting in a relative majority for the adversary, but not in an absolute (adversarial) majority w.r.t. the underlying resource. The adversarial majority advantage can be measured per round by $b_r := n_{\mathsf{a}}^{(r)} - \theta \cdot n_{\mathsf{h}}^{(r)} > 0$ during the attack.

A protocol is said to satisfy self-healing w.r.t. (attack) budget $\mathcal{B}$ provided that there exists a time window $v(\mathcal{B})$ such that, relative to the attack window $[r_l, r_u]$, the protocol satisfies consistency and liveness except possibly during rounds $[r_l - v(\mathcal{B}), r_u + v(\mathcal{B})]$ under the condition that (i) $\sum_{r=r_l}^{r_u} b_r \leq \mathcal{B}$ and (ii) the honest majority condition holds for each round $r' \notin [r_l, r_u]$. This captures that except in a contained region of the execution, the protocol

enjoys the normal consistency and liveness properties. As shown in [2], when different attack regions are sufficiently far apart, this definition applies to an execution with multiple spikes that can be treated in isolation.

3 The Peras Protocol

3.1 Background: Nakamoto-Style PoS (Ouroboros)

We briefly review the Ouroboros proof-of-stake protocol family [28,1,10], which serves as the foundation for our construction.

Slot leadership. Time is divided into discrete *slots*. In each slot, a *leader lottery* independently selects a set of *slot leaders* who are eligible to produce a block. The lottery is stake-weighted: a party's probability of being elected is proportional to its stake, and leadership decisions are independent across parties and slots. In the concrete Ouroboros instantiation, the lottery is implemented via a verifiable random function (VRF) evaluated locally by each party against a stake-dependent threshold; we refer the reader to [1,10] for details. As is common, for the purpose of the security analysis, we assume that the number of honest and adversarial leaders elected in each slot (as a function of their stake) can be well approximated by independent Poisson random variables with parameters r_{h} and r_{a}, respectively [23,24].

What parties do. Each party maintains a *preferred chain*, initialized to the genesis block. The main loop executed by every party in each slot proceeds as follows:

1. *Receive:* Collect all chains (or new blocks) arriving from the network.
2. *Chain selection:* Among all valid chains observed so far, adopt the longest one as the new preferred chain.
3. *Block production:* If the party is a slot leader for the current slot, create a new block extending the preferred chain. The block contains a transaction payload and is signed using a key-evolving signature (KES) scheme to ensure forward and adaptive security. The new chain is then diffused on the peer-to-peer network.

The protocol begins from a genesis block that includes an initial random value (seeding the leadership lottery) and the initial stake distribution.

Epochs and randomness. Multiple slots are grouped into *epochs*, each comprising $\mathsf{epLen} \in \mathbb{N}$ slots, indexed by $j \in \mathbb{N}$. During epoch j, leader election uses the stake distribution $\mathbb{S}_j$ recorded on the blockchain up to g slots before the epoch's start (in [1,10], $g = \mathsf{epLen}$). The epoch randomness for epoch j is derived as a hash of VRF values included in blocks of this epoch up to h slots before the epoch's start (in [1,10], $h = \mathsf{epLen}/3$).

Bootstrapping from genesis. Ouroboros provides a procedure for parties that join the protocol having only knowledge of the genesis block, known as the *Genesis rule* [1]. A joining party listens to the network for sufficiently long and, for any pair of competing chains, applies the following rule: if the chains branch in the recent past, the longer chain is preferred; if they branch further back, preference is given to the chain that is denser (i.e., contains more blocks) in a window of gw slots immediately following the branching point, where gw is a protocol parameter.

Security. Ouroboros, being a longest-chain PoS protocol, was shown [20,12] to provide safety and liveness as long as

$$p_a < \frac{1}{\Delta + 1/p_h}\,, \tag{2}$$

where p_a (resp., p_h) is the probability of a positive number of adversarial (resp., honest) lottery successes appearing in a slot, as determined by the central protocol parameters r_{a} and r_{h}. This is an honest-majority condition in the sense of (1), where the threshold $\theta < 1$ accounts for the fact that some honest leadership opportunities are "wasted" in the Nakamoto race due to network delays.

3.2 Toward Settlement Acceleration

The Ouroboros Peras protocol is an extension of Ouroboros Praos and introduces a voting component designed to expedite settlement in scenarios where there is an optimistic overlap, namely when parties share a lengthy common prefix and their chains diverge only near the end.

Approach. Participants vote on the tip of the chain that results from excluding all blocks that are newer than a specified number of L slots (unless that tip block is too old). If, as a result of this voting, a single block receives more than a designated threshold τ of votes—earning what is termed a "certificate"—that block is awarded additional *weight* $B \gg 1$, a *boost* (where the standard weight of a block is 1). Correspondingly, parties now pick the *heaviest* rather than the longest chain.

The threshold τ is set high enough to prevent multiple blocks from receiving this boost. Specifically, τ is chosen to be $3/4$ of the total votes, based on a standard quorum intersection argument against an attacker controlling a fraction $\alpha < 1/2$ of the stake.[6]

However, in such a regime, honest parties cannot independently produce certificates since the threshold is above $1/2$. Further, even if the threshold were lower, it would be possible for the distribution of honest votes to be such that no single block achieves the threshold with only honest votes. This opens the possibility for the attacker to execute "certificate-up-the-sleeve" attacks by withholding adversarial votes needed for a block to meet the threshold and releasing them only after honest parties have constructed a standard length-B Nakamoto chain that excludes the boosted block. This strategy can lead to two chains of equal weight diverging by weight B. By repeating this tactic, the attacker can create a fork of arbitrary weight, thereby compromising the protocol's consistency.

Cooldown periods. To counteract these risks, the Peras protocol employs a structured voting process with rounds of a fixed length U. If no certificate is generated within a voting round, the protocol initiates a cooldown phase during which voting is temporarily suspended.

During the initial "healing-and-certificate-inclusion" phase of the cooldown period, parties continue with standard Nakamoto block creation until the potential advantage of B that the adversary could gain from a hidden certificate is neutralized. Simultaneously, parties are required to submit the latest certificate they are aware of to the chain. Moreover, there is an age limit for the inclusion of certificates during this phase. In the second phase, parties wait

[6] Throughout this paper, it is assumed that the majority of every committee is honest. This can be approximated (except with negligible probability) by choosing parameters so that committees are large enough.

until the blocks from the first phase have stabilized. Together with the age limit for certificate inclusion, this guarantees that parties use the most recent certificate on the chain as an anchor to achieve consensus on when to resume the voting process, thereby preventing the adversary from exploiting any undisclosed certificates to cause further desynchronization.

Note that there is a tradeoff between the speed of settlement and the duration of the cooldown period. Specifically, the larger the value of B, the quicker the settlement process can potentially occur. However, a larger B also necessitates a longer cooldown period, as the length of the healing phase is directly related to B.

3.3 Protocol Description

The protocol is outlined in Figure 1 and described in detail in the following.

Blocks. The protocol structures time into equal-length intervals known as *slots*. During each slot, participants elect themselves as leaders using the VRF-based local sortition mechanism above. A leader is tasked with extending the heaviest chain they are aware of with a new block B. Each block B is a tuple represented as $\mathsf{B} = (s, \mathsf{P}, h, \mathsf{cert}, \pi, p, \sigma)$, where s indicates the slot index in which the block was produced, P identifies the slot leader who produced the block, h is the hash of the block's parent, cert is a certificate (further explained below), π is the proof of slot leadership, p is the transaction data, and σ is a KES signature by P on the rest of B.

A chain is a non-empty sequence of blocks linked by hash pointers, beginning with a distinguished genesis block G. The *length* of a chain, denoted $\mathsf{len}(C)$, is equal to the number of blocks in the chain, excluding the genesis block. C_{genesis} denotes the unique chain containing only the genesis block G.

Voting rounds. Building on the intuition above, the protocol also segments time into voting rounds (or simply rounds), each comprising U slots. These rounds are sequentially numbered $r = 0, 1, 2, \ldots$, and voting round r corresponds to slots $rU, rU + 1, rU + 2, \ldots, (r + 1)U - 1$ No votes are cast in round 0, which is successful by definition.

The function $\mathrm{rnd}(s) := \lfloor s/U \rfloor$ determines the round to which a given slot s belongs, while $\mathrm{lastSlt}(r) := (r + 1) \cdot U - 1$ denotes the final slot of round r.

Secure committee selection. In each voting round r, a committee $\mathcal{C}_r$ is selected based on a recent stake snapshot $\mathbb{S}$, which can coincide with the Ouroboros epoch snapshot for concreteness. Looking ahead, our security analysis holds w.r.t. any secure committee election mechanism which is adaptively secure and does not over-represent the adversarial resource units in the committee. It is natural to think of a committee as those "coins" of $\mathbb{S}$ that are allowed to cast a vote.

For concreteness, we adopt the VRF-based sortition of Algorand [26]. Each party locally evaluates a VRF (on a domain-separated, round-specific input) to obtain randomness, and uses it to sample from a $\mathrm{Bin}(n, p)$ distribution, where n is the number of stake units the party holds and $p = S/W$, with W the total stake in $\mathbb{S}$ and S a protocol parameter governing the expected committee size. The outcome determines how many votes the party may cast in this round; public verifiability of the VRF ensures that committee membership and voting power can be verified by anyone. We refer to [26,11] for a detailed treatment and for tradeoffs in choosing S.

Protocol **Ouroboros Peras**

Parameters: U: round length; A: certificate expiration age; R: length of chain-ignorance period; K: length of cooldown period; L, D minimal and maximal lookback parameter for voting candidates; T_{CS} time-based settlement parameter.

Variables: The protocol keeps track of the following variables, initialized to the values below:

- $C_{\mathsf{pref}} \leftarrow C_{\mathsf{genesis}}$: preferred chain;
- $\mathcal{C} \leftarrow \{C_{\mathsf{genesis}}\}$: set of chains;
- $\mathcal{V} \leftarrow \emptyset$: set of votes;
- $\mathsf{Certs} \leftarrow \emptyset$: set of certificates;
- $\mathsf{cert}' \leftarrow \mathsf{cert}_{\mathsf{genesis}}$: the latest certificate seen;
- $\mathsf{cert}^* \leftarrow \mathsf{cert}_{\mathsf{genesis}}$: the latest certificate on chain.

Fetching: At the beginning of each slot:

1. Fetch new chains $\mathcal{C}_{\mathsf{new}}$ and votes $\mathcal{V}_{\mathsf{new}}$.
2. Add any new chains in $\mathcal{C}_{\mathsf{new}}$ to $\mathcal{C}$, add any new certificates contained in chains in $\mathcal{C}_{\mathsf{new}}$ to Certs.
3. Add $\mathcal{V}_{\mathsf{new}}$ to $\mathcal{V}$ and turn any new quorum in $\mathcal{V}$ into a certificate cert and add cert to Certs.
4. Set C_{pref} to the heaviest (w.r.t. $\mathsf{Wt}_{\mathsf{P}}(\cdot)$) valid chain in $\mathcal{C}$.
5. Set cert' to the certificate with the highest round number in Certs.
6. Set cert^* to the certificate with the highest round number on C_{pref}.

Block creation: Whenever party P is slot leader in a slot s, belonging to some round r:

1. Create a new block $\mathsf{B} = (s, \mathsf{P}, h, \mathsf{cert}, \pi, p, \sigma)$, where
   - h is the hash of the last block in C_{pref},
   - cert is set to cert' if
     (a) there is no round-$(r-2)$ certificate in Certs, and
     (b) $r - \mathsf{round}(\mathsf{cert}') \leq A$, and
     (c) $\mathsf{round}(\mathsf{cert}') > \mathsf{round}(\mathsf{cert}^*)$,
     and to $\perp$ otherwise,
   - π is the slot-leadership proof,
   - p is a transaction payload, and
   - σ is a signature by P on the rest of B.
2. Extend C_{pref} by B, add the new C_{pref} to $\mathcal{C}$ and diffuse it.

Voting: Party P does the following at the beginning of each voting round r:

1. Let B be the youngest block at least L slots old on C_{pref} but at most D slots old. If no such block exists, exit this procedure.
2. If party P is (voting) committee member in a round r,
   (VR-1A) $\mathsf{round}(\mathsf{cert}') = r - 1$ and cert' was received in the first Δ slots before the end of round $r-1$,
   and
   (VR-1B) B extends the block certified by cert',
   or
   (VR-2A) $r \geq \mathsf{round}(\mathsf{cert}') + R$,
   and
   (VR-2B) $r = \mathsf{round}(\mathsf{cert}^*) + cK$ for some $c > 0$,
   then create a vote $v = (r, \mathsf{P}, h, \pi, \sigma)$, where
   - h is the hash of B,
   - π is the committee-membership proof, and
   - σ is a signature on the rest of v.

   Add v to $\mathcal{V}$ and diffuse it.

Chain output: In each slot, output (C, C_{pref}), where C is obtained as follows:

1. Let $\rho \in \{0,1\}^*$ be a string such that $\rho_i = 1$ if and only if a round-i certificate has been seen (locally).
2. Let ℓ be the maximum slot such that
   (I) $\ell + D$ is followed by at least $\lceil T_{\mathsf{CS}}/B \rceil$ complete rounds i with $\rho_i = 1$
   or
   (II) ℓ is followed by at least $T_{\mathsf{CS}} + K$ complete rounds i with $\rho_i = 0$.
3. C is C_{pref} with all blocks after ℓ pruned.

Fig. 1. The Peras Protocol.

Votes and Certificates. A vote is a tuple $v = (r, \mathsf{P}, h, \pi, \sigma)$, where r is the index of the voting round the vote belongs to, P identifies the voting-committee member casting the vote, h is a hash of the block voted for, π is the committee-membership proof, and σ is a forward-secure signature by P on the rest of v. As already stated above, we omit for simplicity the dependency of r on the concrete stake snapshot relevant for committee election.

A *certificate* is a collection of at least $\tau \cdot S$ votes from a single voting round that all support the same block, where $\tau \in (0, 1]$ is a protocol parameter called the *certificate threshold*. At the beginning of the cooldown phase, a party will include the latest certificate they know of in the blocks they create (unless some other party has already done so). There is an age limit A for certificate inclusion in order to ensure that by the end of the cooldown, parties have agreement on which the latest certificate on the chain is.

The threshold τ is chosen so that the following two properties hold with overwhelming probability (in S) under the honest-majority assumption (1):

1. *Uniqueness:* In any given voting round, certificates cannot be created for two distinct blocks.
2. *Honest support:* Every certificate contains at least one vote cast by an honest committee member.

Both properties follow from a standard arguments: if the adversary controls a fraction $\alpha < 1/2$ of the committee votes, then two certificates for the same round would require at least $2(\tau - \alpha)S$ honest votes, which exceeds the $(1 - \alpha)S$ available honest votes whenever $\tau > (1 + \alpha)/2$. Since $\alpha < 1/2$, it suffices to set $\tau > 3/4$. Similarly, since $\tau > 1/2 > \alpha$, every certificate must contain at least $(\tau - \alpha)S > 0$ honest votes. For a formal treatment of the concentration bounds ensuring that the committee composition reflects the stake distribution with sufficient accuracy, we refer to [26,21].

Chain weight. Each party P assigns a certain weight to every chain C, based on C's length and all certificates that vote for blocks in C that P has seen so far (and thus stored in a local list Certs). Let $\mathsf{certCount}_{\mathsf{P}}(C)$ denote the number of such certificates, i.e.,

$$\mathsf{certCount}_{\mathsf{P}}(C) := |\{\mathsf{cert} \in \mathsf{Certs} : \mathsf{cert} \text{ votes for a block on } C\}| \ .$$

Then, the *weight of the chain* C in P's view is

$$\mathsf{Wt}_{\mathsf{P}}(C) := \mathsf{len}(C) + B \cdot \mathsf{certCount}_{\mathsf{P}}(C)$$

for a protocol parameter B. At any time, the preferred chain of a party is the heaviest chain they know of.

Voting rule. In each voting round r, committee members from $\mathcal{C}_r$ cast a vote for the youngest block B with age between D and L on their preferred chain if and only if the following composite condition is met (if there is no such block, no vote is cast):

$$(\text{VR-1A}) \wedge (\text{VR-1B}) \ \vee \ (\text{VR-2A}) \wedge (\text{VR-2B}) \ ,$$

where

(VR-1A) P has seen a certificate cert_{r-1} for round $r-1$ within the first Δ slots of round $r-1$,

(VR-1B) B extends the block certified by cert_{r-1},

(VR-2A) the last certificate P has seen is from a round at least R rounds back, and

(VR-2B) the last certificate included in P's current chain is from a round exactly cK rounds ago for some integer $c > 0$.

The intuition behind this voting rule is as follows: Rule (VR-1A) serves as the primary criterion for deciding whether an ongoing fast-settlement phase continues in the current round: a committee member participates in the current round only if they have observed a certificate from the previous round in the first Δ slots of that (previous) round.[7] Rule (VR-1B) ensures that certificates from consecutive successful voting rounds are built upon one another, which is essential for accelerating settlement: only if this is the case, will each successful voting round following some block B add the additional weight B to B.

To comprehend the remaining two rules, consider the scenario where the first two conditions are false, indicative of a cooldown period: Rule (VR-2A) prohibits committee members from using the chain to identify an anchor certificate during the initial R rounds of the cooldown, where R is a predefined parameter. This restriction is necessary because the blockchain might be in a state of disarray during this first phase of cooldown, due to a potential certificate up the adversary's sleeve. Finally, Rule (VR-2B) is the actual restart condition: a restart occurs cK rounds after the round number of the latest certificate on chain, for some $c > 0$. The reason for considering values $c > 1$ as well is that there may be restart rounds that do not lead to a certificate (followed immediately by another cooldown), and therefore the certificate with the highest round number on chain might be form $2K$, $3K$, etc. rounds ago.

Chain output. In every slot, parties output their current chain view *pruned back* to the latest block they consider settled given the certificates they have observed. Concretely, let ℓ be the maximum slot such that either (I) $\ell + D$ is followed by sufficiently many complete rounds with certificates so that the combined weight of these certificates guarantees settlement on its own; or (II) ℓ is followed by sufficiently many complete rounds without certificates to conclude settlement based purely on longest-chain behavior. The chain output is C_{pref} with all blocks after ℓ pruned. We consider this conservative settlement rule for simplicity; note that more optimized approaches are possible.

Parameters. The above suggests the following parameterizations for Peras, where T_{HCI} (resp., T_{CS}) is a parameter chosen for the Nakamoto protocol to provide healing from a failed voting round and ensure the presence of at least one honest block (resp., achieve settlement).

- $U \geq 3\Delta / \left(1 - p_a \left(\Delta + \frac{1}{p_h}\right)\right)$ (in slots): voting round length.[8]
- $A = T_{\mathsf{HCI}}$ (in rounds): certificate expiration age.
- $T_{\mathsf{HCI}} \leq R \leq K - 2$ (in rounds): length of chain-ignorance period.
- $K \geq T_{\mathsf{HCI}} + T_{\mathsf{CS}} + 1$ (in rounds): length of cooldown period.

Practical protocol parametrization. The ultimate goal of the protocol is to boost transaction settlement in the sense that a target worst-case probability of a rollback happening to a transaction is achieved quicker than with plain Nakamoto. In plain Nakamoto, the delay is

[7] The main reason for the Δ-requirement is to ensure liveness during fast-settlement phases.

[8] Looking ahead, the lower bound on U ensures that expected honest growth during a subinterval of a round ($U - 3\Delta$ slots) dominates expected adversarial growth during the full round (U slots). For details, see proof of Lemma 5 in Appendix A.8.

corresponds to the time it takes to build sufficiently many blocks on top of the block holding the transaction, which practically means several hours for security-critical application.

In Peras the settlement delay consists of two parts: first, the offset L (to give the system the chance to stabilize on a good candidate block for the voting), and second the number of subsequent boosts required to achieve the target security level. To inform practice about the various tradeoffs possible here, a tool has been developed [34] to calculate settlement errors for various choices. In terms of settlement delay, one can observe that already for quite conservative choices of L in the order of five minutes, U in the order of two minutes, and a boost weight of 50, the rollback probability is in the order of 10^{-50} even after a single boost and against a 10% adversary. At the same time, the probability of entering a cooldown with this choice of L (due to the system not stabilizing on a block that is at least L old) is less than one in a million, meaning that a cooldown would be expected once every three years (given the above cadence of voting rounds). This reliably brings down the settlement time from hours to tens of minutes with our approach.

4 Proof Overview

This section gives a high-level roadmap of the security argument; the full details can be found in Appendix A. The proof follows the blocktree-based analysis paradigm for longest-chain protocols and its PoS instantiations, centered around the notions of *reach* and *margin* (see below), as developed in [28] and subsequent work (e.g., [4,24]) on tight (linear) consistency/settlement bounds. The proofs in this work extend this analytic toolkit to include Peras certificates. The consistency and liveness proofs proceed along the following lines:

1. *Modeling executions:* We formalize what can happen in an execution via a *characteristic string*, which consists of a *leader string*, describing the block-leader schedule and—which is novel in the Peras context—a *voting string* $\sigma \in \{0, ?, 1\}^*$, describing whether the parties observe a successful certificate round (1), an ambiguous round (?), or a no-vote/no-certificate round (0). Based on this, we axiomatically define the notion of a *blocktree with certificates* compatible with a given characteristic string.
2. *Reach and margin:* We measure adversarial power by defining two quantities extracted from these blocktrees: reach ρ—hidden weight that could surface later—and (relative) margin μ_ℓ—how feasible it is to maintain two competing *dominant* branches (branches heavy enough that an honest party could adopt them) that disagree before a fixed slot ℓ; negative margin μ_ℓ means this is not possible.
3. *Proving recurrences:* We show how reach and margin evolve across slots depending on the characteristic string, and in particular on the voting string. These recurrences are the technical engine for tracking security over time.
4. *Control over voting string:* An important assumption made by the above recurrence-based tracking is that the voting string has a particular "cyclic" structure. We prove that under a suitable high-probability event on the leader string (a *good leader string*), this is indeed the case via an inductive argument.
5. *Concluding consistency and liveness:* Finally, we show that the above structure forces margin relative to the protocol's pruning point to become negative and stay negative (yielding settlement/consistency), and that within a bounded time window, an honest block must appear on all dominant chains (yielding liveness).

Sections 4.1 to 4.5 below elaborate on each of these steps. The lifting to dynamic stake and the self-healing argument are discussed in Sections 4.6 and 4.7, respectively.

4.1 Modeling Executions

Leader and voting strings (Appendix A.1). As in prior longest-chain analyses, the *leader string* $w = w_1 \cdots w_N \in (\mathbb{N} \times \mathbb{N})^N$ records, per slot, how many honest and adversarial leaders are eligible to extend the chain. Compared to previous work [28,1,20], our setting also involves the fast-settlement voting overlay, which is captured by an additional *voting string* $\sigma = \sigma_1\sigma_2 \cdots \in \{0, ?, 1\}^*$, where round i is labeled

- 1 if some honest party sees a round-i certificate by within the first Δ slots of round i,
- ? if no such certificate is seen but at least one honest party votes in round i,
- 0 otherwise.

The pair (w, σ) is referred to as the *characteristic string*. Note that one symbol of σ covers U symbols of w.

Blocktrees (Appendix A.2). Given an execution (w, σ), a *blocktree with certificates* (Definition 2 and Definition 10) is a rooted directed tree whose vertices represent blocks. Each vertex carries: (i) its slot index, (ii) the set of rounds in which it is certified, and (iii) whether it was created by an honest or adversarial leader. The axioms enforce that the tree is consistent with the protocol and network timing. Informally, the most important constraints are:

- *Time consistency:* Along any chain, slot labels strictly increase, certificate labels strictly increase, and a certificate for round r can only label blocks from sufficiently early slots (no "future" certificates).
- *Count constraints:* The leader string fixes the number of honest blocks per slot; the voting string fixes whether a certificate can exist in a round (none for 0-rounds, at most one for ?-rounds, exactly one for concluded 1-rounds).
- *Protocol-respecting placement:* Honest block creators extend a maximum-weight chain in their local view; a certified block must lie on a maximum-weight chain in the local view at the time of voting, must be "recent" (within D), and consecutive certificates must stack on a single chain.

This abstraction lets us reason about all adversarial message scheduling and withheld information by taking worst cases over trees consistent with (w, σ).

4.2 Reach and Margin

The main analytical quantities are so-called reach and margin (cf. Appendix A.3). In order to pin them down precisely, first define a "public" part of the execution, capturing the information that is inevitably known to all honest parties at a given time, the *public subtree* $\overline{F}$. Next, define the *advantage* of a chain T as its weight relative to this public baseline, $\alpha_F(T) = \mathsf{wt}_F(T) - \mathsf{wt}(\overline{F})$. Reach is the maximum possible advantage:

$$\rho(F) = \max_T \alpha_F(T) \qquad \text{and} \qquad \rho(w, \sigma) = \max_{F \vdash (w,\sigma)} \rho(F)\,,$$

where the maximum is over all trees F compatible with (w, σ), denoted by $F \vdash (w, \sigma)$. Intuitively, reach upper-bounds how much "hidden" (adversarial) weight could exist beyond what honest parties consider public.

For the remainder of this discussion, fix some slot $\ell \geq 1$. *Margin relative to* ℓ measures disagreement before slot ℓ:

$$\mu_\ell(F) = \max_{T_1 \not\preccurlyeq_\ell T_2} \min\{\alpha_F(T_1), \alpha_F(T_2)\}\,, \qquad \mu_\ell(w, \sigma) = \max_{F \vdash (w,\sigma)} \mu_\ell(F)\,.$$

Here $T_1 \not\sim_\ell T_2$ means that the two chains do not share a common prefix through slot ℓ (i.e., they disagree on the ledger state before ℓ). When the margin relative to ℓ stays negative, there cannot exist two *dominant* chains (chains heavy enough to be held by honest parties) that disagree before ℓ; equivalently, all chains that honest parties may adopt lie on a single branch through ℓ. This is precisely the condition needed to argue settlement of blocks up to ℓ (cf. Appendix A.4).

4.3 Proving Recurrences

The next step is to derive inequalities that allow to track reach and margin across the execution. The idea is to take an execution up to a certain point, to consider an extension of the execution, and then to derive bounds on reach and margin after the extension in terms of their values before the extension. The key cases are:

- 1-*rounds (happy periods):* The main insight in this case is that margin decreases by B for every certificate that appears during the happy periods, which leads to accelerated settlement. See Appendix A.5 for additional details.
- 0-*rounds (cooldowns):* When no certificates appear, the system behaves like longest-chain PoS without the voting overlay. In this regime, certificates do not contribute any weight, and the only way for reach to grow is via additional blocks. The inequalities for this case follow those in previous work [24]. For details, consult Appendix A.6.
- ?-*rounds (transitions):* The fact that some honest parties voted but no certificates were seen implies that the adversary may be able to complete said votes to a certificate at any future point of its choosing. Thus, reach and margin increase by (approximately) B. See Appendix A.7 for details.

These bounds are designed to compose along the (structured) voting string.

4.4 Control over Voting String

The recurrences above assume that the *voting string* follows a constrained format that alternates between: (i) *happy periods* of consecutive 1-rounds (the optimistic fast-settlement path), and (ii) *cooldowns* consisting of a short ?-transition followed by many 0-rounds (during which the protocol reverts to longest-chain behavior and includes/settles the latest certificate). This is proved using an inductive argument. See Appendix A.9 for details.

Good leader strings (GLS) and tracking. The proof of the voting-string format relies on probabilistic properties of the leader string w that hold with overwhelming probability. Section A.8 encapsulates these into the definition of a good leader string and shows that bad leader strings occur with negligible probability (Lemma 5). At a high level, GLS supplies three kinds of guarantees during cooldown:

- *Reach control:* During the healing/certificate-inclusion periods of the cooldown, reach recovers from the "damage" from the ?-round and is bounded by the end of it; this way it can be used as starting point for tracking margin during the certificate-settlement portion of the cooldown (see third item).
- *Chain-quality-like guarantees:* At least one honest block appears in the healing/certificate-inclusion period of the cooldown. This ensures that the latest certificate can be successfully included in the chain before it expires.[9]

[9] The chain-quality-like guarantees also ensure that there is liveness during both happy periods and cooldowns.

– *Margin decay and persistence:* Over the certificate-settlement portion of a cooldown, GLS ensures that the margin relative to the end of the healing/ certificate-inclusion portion becomes negative and then remains negative, which implies agreement on the ledger up to that point, and in particular on the certificate acting as restart anchor.

4.5 Concluding Consistency and Liveness

For consistency (Appendix A.10), it remains to appropriately track reach and margin to show that relative to some pruning slot, margin becomes and remains negative. Liveness (Appendix A.11) is argued based on the chain-quality-like guarantees of the GLS property.

4.6 Dynamic Stake and Bootstrapping from Genesis

The analysis up to this point considers a static stake distribution. As in prior Ouroboros work [1,10], this can be lifted to dynamic stake via an inductive epoch structure, where each epoch derives its stake distribution and randomness from sufficiently stabilized portions of the chain. Because the key analytic quantities—reach and margin (cf. Definition 13)—have been re-established for our weight-based setting, the same lifting applies: the epoch is divided into three phases that, respectively, stabilize the stake distribution, ensure an honest randomness contribution, and stabilize the randomness-determining blocks.

Similarly, bootstrapping from the genesis block via the Ouroboros Genesis rule [1] carries over by replacing density with weight in the chain-comparison rule for newly joining parties (see Appendix A.12 for details).

4.7 Self-Healing

Self-healing (cf. Appendix A.13 for details) considers executions in which, for a bounded spike budget $\mathcal{B}$, reduced honest participation temporarily gives the adversary an effective majority, after which the honest-majority condition is restored. Since the spike stems from absent honest parties rather than additional adversarial stake, voting committees are never adversarially dominated (with overwhelming probability). The protocol can be hardened against such spikes by increasing the cooldown parameters to $T'_{\mathsf{HCI}} = \Theta(T_{\mathsf{HCI}} + \mathcal{B})$ and $T'_{\mathsf{CS}} = \Theta(T_{\mathsf{CS}} + \mathcal{B})$. If honest majority is restored during a cooldown, the stretched parameters provide enough runway to re-establish certificate inclusion and settlement via the 0-round recurrence as in Lemma 6. If it is restored during a 1-round streak, the 1-round recurrence drives margin down until either the safe regime is reached or a cooldown is entered, reducing to the previous case. In either case, after a recovery period proportional to $\mathcal{B}$, the protocol returns to its baseline consistency and liveness guarantees.

References

1. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 913–930. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243848
2. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Consensus redux: Distributed ledgers in the face of adversarial supremacy. Cryptology ePrint Archive, Paper 2020/1021 (2020), https://eprint.iacr.org/2020/1021, computer Security Foundation (CSF) 2024, to appear

3. Bagaria, V.K., Kannan, S., Tse, D., Fanti, G., Viswanath, P.: Prism: Deconstructing the blockchain to approach physical limits. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 585–602. ACM (2019). https://doi.org/10.1145/3319535.3363213, https://doi.org/10.1145/3319535.3363213
4. Blum, E., Kiayias, A., Moore, C., Quader, S., Russell, A.: The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In: Chawla, S. (ed.) 31st SODA. pp. 1135–1154. ACM-SIAM (Jan 2020). https://doi.org/10.1137/1.9781611975994.69
5. Buterin, V., Griffith, V.: Casper the friendly finality gadget (2019)
6. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance. In: 3rd Symposium on Operating Systems Design and Implementation (OSDI 99). USENIX Association, New Orleans, LA (Feb 1999)
7. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155–183 (2019), in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I
8. D'Amato, F., Neu, J., Tas, E.N., Tse, D.: Goldfish: No more attacks on ethereum?! Cryptology ePrint Archive, Paper 2022/1171 (2022), https://eprint.iacr.org/2022/1171, financial Cryptography 2024, to appear
9. DAmato, F., Zanolini, L.: Recent Latest Message Driven GHOST: Balancing Dynamic Availability With Asynchrony Resilience . In: 2024 IEEE 37th Computer Security Foundations Symposium (CSF). pp. 127–142. IEEE Computer Society, Los Alamitos, CA, USA (Jul 2024). https://doi.org/10.1109/CSF61375.2024.00001, https://doi.ieeecomputersociety.org/10.1109/CSF61375.2024.00001
10. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 66–98. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_3
11. David, B., Magri, B., Matt, C., Nielsen, J.B., Tschudi, D.: GearBox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 683–696. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3559375
12. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a race and nakamoto always wins. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 859–878. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417290
13. Dinsdale-Young, T., Magri, B., Matt, C., Nielsen, J.B., Tschudi, D.: Afgjort: A partially synchronous finality layer for blockchains. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 24–44. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_2
14. Feldman, P., Micali, S.: An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. SIAM Journal on Computing **26**(4), 873–933 (1997)
15. Fitzi, M., Gazi, P., Kiayias, A., Russell, A.: Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. IACR Cryptol. ePrint Arch. p. 1119 (2018), https://eprint.iacr.org/2018/1119
16. Fitzi, M., Gazi, P., Kiayias, A., Russell, A.: Ledger combiners for fast settlement. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12550, pp. 322–352. Springer (2020). https://doi.org/10.1007/978-3-030-64375-1_12, https://doi.org/10.1007/978-3-030-64375-1_12
17. Garay, J., Kiayias, A., Shen, Y.: Proof-of-work-based consensus in expected-constant time. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 96–125. Springer Nature Switzerland, Cham (2024)
18. Garay, J., Kiayias, A., Shen, Y.: State machine replication among strangers, fast and self-sufficient. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025. pp. 3–36. Springer Nature Switzerland, Cham (2025)

19. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_10
20. Gazi, P., Kiayias, A., Russell, A.: Tight consistency bounds for bitcoin. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 819–838. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3423365
21. Gaži, P., Kiayias, A., Russell, A.: Fait accompli committee selection: Improving the size-security tradeoff of stake-based committees. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 845–858. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM (Nov 2023), 30th ACM SIGSAC Conference on Computer and Communications Security , CCS 2023 ; Conference date: 26-11-2023 Through 30-11-2023
22. Gaži, P., Motaqy, Z., Russell, A.: A tight analysis of GHOST consistency. Cryptology ePrint Archive, Paper 2024/1830 (2024), https://eprint.iacr.org/2024/1830
23. Gazi, P., Ren, L., Russell, A.: Practical settlement bounds for proof-of-work blockchains. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1217–1230. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3559368
24. Gazi, P., Ren, L., Russell, A.: Practical settlement bounds for longest-chain consensus. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 107–138. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_4
25. Gelashvili, R., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A., Xiang, Z.: Jolteon and Ditto: Network-Adaptive Efficient Consensus with Asynchronous Fallback. In: Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers. pp. 296–315. Springer-Verlag, Berlin, Heidelberg (2022)
26. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. p. 51–68. SOSP '17, Association for Computing Machinery, New York, NY, USA (2017)
27. Kiayias, A.: Ouroboros: self-healing in the wild. IOG Blog (Dec 2025), https://www.iog.io/news/ouroboros-self-healing-in-the-wild
28. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 357–388. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_12
29. Malkhi, D., Momose, A., Ren, L.: Towards practical sleepy bft. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 490–503. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3623073, https://doi.org/10.1145/3576915.3623073
30. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. White paper (2008), https://bitcoin.org/bitcoin.pdf
31. Neu, J., Tas, E.N., Tse, D.: Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In: 2021 IEEE Symposium on Security and Privacy. pp. 446–465. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00045
32. Pass, R., Shi, E.: Hybrid consensus: Efficient consensus in the permissionless model. In: International Symposium on Distributed Computing (2016), https://api.semanticscholar.org/CorpusID:1171104
33. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 3–33. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_1
34. Peras probability calculator (2025), https://peras.cardano-scaling.org/dashboard/index.html
35. Stewart, A., Kokoris-Kogia, E.: Grandpa: a byzantine finality gadget (2020)

A Protocol Analysis

A.1 Lotteries and Characteristic Strings

Two independent randomized party-selection procedures play important roles in our protocol; we call them *lotteries*. The first lottery, referred to as the *leader lottery*, selects for each slot a collection of *slot leaders* that are allowed to create a block in that slot. The second lottery, called the *voter lottery*, selects committee members (i.e., *voters*) for each voting round. Both of these lotteries are implemented using standard private sortition methods [7]. The leader lottery is parameterized so that the expected number of elected slot leaders is a small constant and, in particular, is relatively likely to generate a slot with a single leader; this determines the dynamics of the Nakamoto consensus process. The voter lottery is parameterized to generate a larger committee used for the fast settlement process.

We use so-called *characteristic strings* to indicate a summary of the outcomes of these two independent lotteries. In particular, we use a *leader string* and a *voting string* to record the outcomes of the leader and the voter lottery, respectively; as detailed below.

Leader string. The leader lottery is reflected by the *leader string* over the alphabet $\Sigma = \mathbb{N}\times\mathbb{N}$; specifically, an execution over N slots gives rise to a leader string $w = w_1 \dots w_N \in \Sigma^N$ where, intuitively, each symbol $w_i = (h_i, a_i) \in \Sigma$ indicates that h_i honest parties and a_i adversarial parties were eligible slot leaders for slot i.

For a leader string $w = w_1 \dots w_n \in \Sigma^n$ where each $w_i = (h_i, a_i) \in \mathbb{N} \times \mathbb{N}$, we define $\#_{\mathsf{h}}(w) := \sum_{i=1}^{n} h_i$ and similarly $\#_{\mathsf{a}}(w) := \sum_{i=1}^{n} a_i$, i.e., the total number of honest and adversarial slot leaders over a sequence of slots corresponding to w. Moreover, we sometimes make use of a similar quantity $\#_{[\mathsf{a}]}(w)$ that denotes the number of symbols in w with positive second coordinate, i.e.,

$$\#_{[\mathsf{a}]}(w) := |\{i \in \{1, \dots, n\} \,:\, w_i \notin \mathbb{N} \times \{0\}\}| \ .$$

Similarly, let

$$\#_{[\mathsf{ha}]}(w) := |\{i \in \{1, \dots, n\} \,:\, w_i \neq (0, 0)\}| \ .$$

Leader string distribution. In our executions, leader strings arise from a distribution $\mathcal{D}(r_{\mathsf{h}}, r_{\mathsf{a}})$ defined as follows: for each slot i and $w_i = (h_i, a_i)$, the random variables h_i and a_i are independent Poisson-distributed random variables with rates r_{h} and r_{a}, respectively. Throughout the analysis we assume that the parameters defining $\mathcal{D}$ satisfy (2).

Voting string. The outcome of the voter lottery is captured by a *voting string*. A voting string is a string $\sigma = \sigma_1\sigma_2 \dots \in \{0, ?, 1\}^*$, with implicit understanding that $\sigma_0 = 1$ represents the genesis round, and for $i \geq 1$,

$$\sigma_i \ = \ \begin{cases} 1 & \text{if at least one hoenst party saw a round-}i\text{ certificate within the first } \Delta \\ & \text{slots of round } i, \\ ? & \text{else if at least one honest party voted in round } i, \\ 0 & \text{otherwise.} \end{cases}$$

We sometimes refer to a round i as an s-round if $\sigma_i = s$, for any $s \in \{1, ?, 0\}$.

We refer to leader strings and voting strings jointly as characteristic strings.

Executions. We say that a leader string w and a voting string σ have *consistent duration* if $\mathrm{rnd}(|w|) = |\sigma|$, and in such case we call the pair (w, σ) an *execution* and $|w|$ is its *length in slots*. (Notice that the length of an execution in slots does not need to be an integer multiple of U.) For an execution (w, σ) with $|w| = N$ and an index $i \in \{0, \ldots, N\}$, we denote by

$$(w, \sigma)_{i\rceil} := (w_{i\rceil}, \sigma_{\mathrm{rnd}(i)\rceil})$$

the *prefix execution* covering the initial i slots of (w, σ). Conversely, we call the execution (w, σ) an *extension* of the execution $(w, \sigma)_{i\rceil}$.

A.2 Blocktrees with certificates

We now define a tree-shaped structure to capture important aspects of the execution of our protocol. For a vertex v in a tree F, let $\mathsf{desc}_F(v)$ denote the set of all descendants (direct and indirect) of v excluding v itself, and $\mathsf{desc}^*_F(v) := \mathsf{desc}_F(v) \cup \{v\}$.

Definition 2 (Blocktree with certificates). *Let (w, σ) be an execution with $|w| = N$ and $|\sigma| = M$, i.e., $\mathrm{rnd}(N) = M$. A* blocktree with certificates *for an execution (w, σ) is a directed, rooted tree $F = (V, E)$ with three labeling functions:*

- $\mathsf{l}_{\#} : V \to \{0, \ldots, N\}$ *where $\mathsf{l}_{\#}(v)$ is called the* slot label *of v and represents the slot in which the corresponding block was created;*
- $\mathsf{l}_{\mathsf{c}} : V \to 2^{\{1,\ldots,M\}}$ *where $\mathsf{l}_{\mathsf{c}}(v)$ is called the* certificate label *of v and records the set of voting rounds in which the block corresponding to v was certified;*
- $\mathsf{l}_{\mathsf{type}} : V \to \{\mathsf{h}, \mathsf{a}\}$ *where $\mathsf{l}_{\mathsf{type}}(v)$ is referred to as the* type *of the vertex: when $\mathsf{l}_{\mathsf{type}}(v) = \mathsf{h}$, we say that the vertex is* honest*; otherwise it is* adversarial*.*

Edges are directed "away from" the root so that there is a unique directed path from the root to any vertex, and the tree must satisfy the following axioms:

Time consistency:

(T1) the root $r \in V$ is honest and is the only vertex with slot label $\mathsf{l}_{\#}(r) = 0$;

(T2) slot and certificate labels along any directed path are strictly increasing: for each $v \in V$ and $v' \in \mathsf{desc}_F(v)$, we have $\mathsf{l}_{\#}(v) < \mathsf{l}_{\#}(v')$ and $c \in \mathsf{l}_{\mathsf{c}}(v) \wedge c' \in \mathsf{l}_{\mathsf{c}}(v') \Rightarrow c < c'$;

(T3) for each $v \in V$ and $r \in \mathsf{l}_{\mathsf{c}}(v)$, we have $\mathsf{l}_{\#}(v) \leq \mathrm{lastSlt}(r-1)$;

Block and certificate counts:

(C1) if $w_i = (h_i, a_i)$ then there are exactly h_i honest vertices of F with the slot label i and if the number of adversarial vertices with slot label i is nonzero then $a_i > 0$;

(C2) if $\sigma_r = 0$ then there is no $v \in V$ with $r \in \mathsf{l}_{\mathsf{c}}(v)$, otherwise there is at most one such $v \in V$;

(C3) if $\sigma_r = 1$ and $\mathrm{lastSlt}(r-1) + 1 + \Delta \leq N$ then $\exists v \in V : r \in \mathsf{l}_{\mathsf{c}}(v)$.

We write $F \vdash (w, \sigma)$ to indicate that F is a blocktree with certificates for an execution (w, σ). As notational shorthands, we define

$$\mathcal{H}(F) := \{v \in V \,:\, \mathsf{l}_{\mathsf{type}}(V) = \mathsf{h}\} \text{ and}$$
$$\mathcal{C}(F) := \{r \in [M] \,:\, \exists v \in V \text{ such that } r \in \mathsf{l}_{\mathsf{c}}(v)\}$$

to refer to the sets of all honest vertices and all rounds that produced a certificate, respectively. For simplicity of notation, we assume $\mathcal{H}(F) \cap \mathcal{C}(F) = \emptyset$. We write $\mathcal{H}$ (resp. $\mathcal{C}$) when F is clear from the context. We sometimes refer to an index $c \in \mathcal{C}$ as a certificate; this is justified as each round produces at most one certificate (C2). We talk about a ?-certicate (resp., 1-certificate) if $\sigma(c) =$? (resp., $\sigma(c) = 1$).

We will refer to blocktrees with certificates simply as *trees* when the context is clear. Unless explicitly stated otherwise, in the rest of the paper we reserve the term "tree" for the above structure, as opposed to the underlying graph-theoretic notion.

It is easy to see the correspondence between the above axioms and the constraints imposed in the protocol execution. In particular, axiom (T1) postulates the existence of the genesis block; axioms (T2)–(T3) guarantee the time-consistency of the execution, namely that vertices and certificates only appear on top of earlier vertices and certificates. Axiom (C1) captures that an honest party uses a leader-lottery success to produce exactly one block, while the adversary might use it to produce arbitrarily many blocks (or none at all). Similarly, axioms (C2)–(C3) maintain that the blocktree contains the correct number of certificates: none for 0-rounds, at most one for a ?-round, and exactly one for any concluded 1-round.

Looking ahead, we will formalize additional properties of blocktrees—that are guaranteed to be satisfied by our protocol's execution—in Definition 10. Towards that, we need to establish some necessary notation.

Definition 3 (Certified vertices). *A vertex v is called* certified *if $\mathsf{I}_\mathsf{c}(v) \neq \emptyset$, and in particular v is called* 1-certified *if $\exists r \in \mathsf{I}_\mathsf{c}(v)\colon \sigma_r = 1$.*

Definition 4 (Subtrees and restrictions). *Let $F \vdash (w, \sigma)$, let execution (w', σ') be an extension of (w, σ) and $F' \vdash (w', \sigma')$. We say that F is a* subtree *of F', denoted $F \sqsubseteq F'$, if F is a subgraph of F' satisfying that for each $v \in F$:*

(i) v's $\mathsf{I}_\#$- and I_type-labels are identical in F and F'; and

(ii) v's I_c-label in F is a subset of its I_c-label in F';

In particular, a subtree F is called a restriction *of F' to a slot s, denoted $F_{s\rceil}$, if*

(i) F contains exactly all vertices $v \in F'$ such that $\mathsf{I}_\#(v) \leq s$;

(ii) for each $v \in F$, v's I_c-label in F is the intersection of its I_c-label in F' with the set $[s]$.

An individual blockchain constructed during the protocol execution is represented by the notion of a *chain*, defined next.

Definition 5 (Chains). *A path in a tree F originating at the root is called a* chain *(note that a chain does not necessarily terminate at a leaf). As there is a one-to-one correspondence between directed paths from the root and vertices of a tree, we routinely overload notation so that it applies to both chains and vertices. Specifically, we let $\mathsf{len}(T)$ denote the* length *of the chain, equal to the number of edges on the path; hence $\mathsf{len}(v)$ also denotes the depth of a vertex. We sometimes emphasize the tree F from which v is drawn by writing $\mathsf{len}_F(v)$. Likewise, we let $\mathsf{I}_\#(\cdot)$ apply to chains by defining $\mathsf{I}_\#(T) := \mathsf{I}_\#(v)$, where v is the terminal vertex on the chain T. We say that a chain is* honest *if the last vertex of the chain is honest.*

Definition 6 (Weight function). *For a tree F and a chain T in F we define the* weight *of T in F as*

$$\mathsf{wt}_F(T) := \mathsf{len}_F(T) + B \cdot \sum_{u \in T} |\mathsf{I}_\mathsf{c}(u)| \ ,$$

where the sum goes over all vertices of $u \in T$. We overload the notation and define

$$\mathsf{wt}(F) := \max_{T \ \mathit{chain\ in}\ F} \mathsf{wt}_F(T) \ .$$

Note that the weight function $\mathsf{wt}(\cdot)$ is intended to model the function $\mathsf{Wt}(\cdot)$ used in our protocol (cf. Section 3), we maintain the notational distinction for clarity.

In the execution of our protocol, each action of an honest party—creating a block or casting a vote—is implicitly *justified* by that honest party's view: the set of blocks and certificates it is aware of at the moment of taking the action. Intuitively, the honest party's action must be consistent with this view: it creates a block by extending the heaviest chain it sees, or votes for a block lying on this chain. We make this notion of a justification explicit in the following definition, and formulate the implied constraints as axioms in Def. 10, after we define all the necessary tools to express them.

Definition 7 (Justifications). *Given a blocktree with certificates $F \vdash (w, \sigma)$ and a slot $s \in \{1, \ldots, |w|\}$, a* justification *J for slot s in F is a subtree of the restriction of F to slot $s-1$, i.e., $J \sqsubseteq F_{s-1\rceil}$. When F is clear from the context, we write $\mathrm{jslot}(J) = s$ to denote the slot for which J is a justification. We say that F is a* tree with justifications *if it is equipped with two sets of justifications $(J_v)_{v \in \mathcal{H}(F)}$ and $(J_c)_{c \in \mathcal{C}(F)}$, where*

- *for each $v \in \mathcal{H}(F)$, J_v is a justification for slot $\mathsf{l}_{\#}(v)$ in F; and*
- *for each $c \in \mathcal{C}(F)$, J_c is a justification for slot $\mathrm{lastSlt}(c-1)+1$ in F.*

Definition 8 (Propagation). *Consider a tree $F \vdash (w, \sigma)$ with justifications $(J_v)_{v \in \mathcal{H}}$ and $(J_c)_{c \in \mathcal{C}}$. We say that:*

- *a justification J is* propagated *in F if $\mathrm{jslot}(J) \leq |w| - \Delta$;*
- *a 1-certificate $c \in \mathcal{C}(F)$ is* propagated *in F if $\mathrm{lastSlt}(c-1)+1 \leq |w| - \Delta$.*

Definition 9 (Public subtree). *Consider a tree $F \vdash (w, \sigma)$ with justifications $(J_v)_{v \in \mathcal{H}}$ and $(J_c)_{c \in \mathcal{C}}$. We denote by $\overline{F}$ the* publicly known *(or simply* public*) subtree of F, which is obtained by the following procedure:*

(i) Remove all vertices v with $\mathsf{l}_{\#}(v) > |w| - \Delta$ and adjacent edges.

(ii) For each remaining v, remove from $\mathsf{l}_{\mathsf{c}}(v)$ all ?-certificates and all unpropagated 1-certificates.

(iii) Iteratively remove all adversarial leaves (and adjacent edges) that are not 1-certified.

(iv) Add back all vertices, edges, and l_{c}-labels that appear in any propagated justification.

Definition 10 (Protocol-respecting trees). *A blocktree $F \vdash (w, \sigma)$ with justifications $(J_v)_{v \in \mathcal{H}}$ and $(J_c)_{c \in \mathcal{C}}$ is called* protocol-respecting *if it additionally satisfies the following axioms:*

Block and certificate placement:

(P1) Each justification J satisfies $\overline{F_{(\mathrm{jslot}(J)-1)\rceil}} \sqsubseteq J$.

(P2) Each honest vertex v extends a maximum-weight chain in J_v.

(P3) For any $v \in V$ and any $c \in \mathsf{l}_{\mathsf{c}}(v)$, v lies on a maximum-weight chain in J_c; and moreover, $\mathsf{l}_{\#}(v) \geq \mathrm{lastSlt}(c-1) - D$.

(P4) Any $u, v \in V$ such that $\exists r \colon r \in \mathsf{l}_{\mathsf{c}}(u) \wedge r+1 \in \mathsf{l}_{\mathsf{c}}(v)$ satisfy $v \in \mathsf{desc}^*_F(u)$.

The axioms again have an intuitive interpretation with respect to our protocol: (P1) requires that any justification—capturing the view of an honest block creator or an honest voter—must contain the whole public subtree at that time. (P2) captures that honest block creators extend a block that in their view (described by J_v) extends a maximum-weight chain; (P3) postulates that any certified block must lie on a maximum-weight chain in some honest view at the time of voting (described by J_c), and must be 'recent' on this

chain as enforced by the protocol; and finally (P4) guarantees that certificates coming from consecutive rounds "extend one another", i.e., lie on a single chain.

In the rest of the paper, we will only be considering protocol-respecting trees (and often referring to them as trees); the reason we defined this notion separately is to take advantage of Definitions 3–9.

Definition 11 (Dominant chains). *Let T be a chain in a tree F. We call T* dominant *in F if $\mathsf{wt}_F(T) \geq \mathsf{wt}(\overline{F})$.*

Definition 12 (Branches). *For an integer $\ell \geq 1$ and for two chains T and T' of a tree F, we write $T \sim_\ell T'$ if the two chains share a vertex with a $\mathsf{I}_\#$-label greater than or equal to ℓ. The set of all chains $T' \in F$ such that $T \sim_\ell T'$ is called the* branch *of T in F and denoted $\mathsf{B}_F(T; \ell)$; when ℓ can be inferred from context, we write $\mathsf{B}_F(T)$.*

Intuitively, $T \sim_\ell T'$ guarantees that the respective blockchains agree on the state of the ledger up to time slot ℓ. Looking ahead, the adversary can make two honest parties disagree on the state of the ledger up to time ℓ only if she makes them hold two chains $T \not\sim_\ell T'$.

A.3 Analytic Quantities: Reach (ρ) and Margin (μ_ℓ)

Definition 13 (Advantage, reach, margin). *For a tree $F \vdash (w, \sigma)$, we define the* advantage *of a chain $T \in F$ as*

$$\alpha_F(T) = \mathsf{wt}_F(T) - \mathsf{wt}(\overline{F}) \; ;$$

in particular, a chain T is dominant in F if and only if $\alpha_F(T) \geq 0$. We then define

$$\rho(F) := \max_{T \text{ in } F} \alpha_F(T) \qquad \text{and} \qquad \rho(w, \sigma) := \max_{F \vdash (w,\sigma)} \rho(F) \; ,$$

and in both cases we refer to the quantity $\rho(\cdot)$ as reach *(of F and the pair (w, σ), respectively). For a given pair (w, σ), we sometimes refer to a tree F and a chain T maximizing the above expressions as a* witness tree *and a* witness chain*, respectively; note that these are not necessarily unique.*

Finally, we define the margin *of F, denoted $\mu_\ell(F)$, to be the "penultimate" advantage taken over chains T_1, T_2 of F such that $T_1 \not\sim_\ell T_2$:*

$$\mu_\ell(F) := \max_{T_1 \not\sim_\ell T_2} \Big(\min\{\alpha_F(T_1), \alpha_F(T_2)\} \Big) \, .$$

There might exist multiple such pairs in F, but under the condition $\ell \geq 1$ there will always exist at least one such pair, as the trivial chain T_0 containing only the root vertex satisfies $T_0 \not\sim_\ell T$ for any T and $\ell \geq 1$, in particular $T_0 \not\sim_\ell T_0$. For this reason, we will always consider $\mu_\ell(\cdot)$ only for $\ell \geq 1$. We again overload the notation by defining

$$\mu_\ell(w, \sigma) := \max_{F \vdash (w,\sigma)} \mu_\ell(F) \, .$$

We use the terms witness tree and witness chains analogously also in the case of margin, it will be always clear from the context whether we are referring to witnesses with respect to reach or margin.

In the analysis we make use of the *honest depth* quantity introduced in [24]. Intuitively, the honest depth $h_\Delta(x)$ of a string x captures the minimum growth of honest blockchains over a period of slots corresponding to x, in the absence of any certificates. More concretely, it is the minimum number of times during x that an honest slot leader must create a block at a higher depth because it is guaranteed to "see" an honest blockchain at one depth lower that was created at least Δ slots earlier.

Definition 14 (Honest depth h_Δ). *For $x \in \{0,1\}^*$, we define $h_\Delta(x)$ inductively so that $h_\Delta(\epsilon) = 0$, $h_\Delta(x0) = h_\Delta(x)$, and $h_\Delta(x1) = h_\Delta(x_{\lceil \Delta}) + 1$. We in fact overload h_Δ to apply to strings from $\Sigma^* = (\mathbb{N} \times \mathbb{N})^*$, in which case symbols with non-zero first coordinate (i.e., from $((\mathbb{N} \setminus \{0\}) \times \mathbb{N})^*$ are counted as 1s, while symbols from $(\{0\} \times \mathbb{N})^*$ are treated as 0s.*

A.4 Margin and Consistency

Intuitively, there is a natural connection between margin and settlement, which was formally established for longest-chain protocols (for the appropriate definition of margin) in [28,20]; we extend it to our setting with certificates below. This motivates our effort to upper-bound μ_ℓ. (Cf. Section 2 for a definition of prune$(\cdot)$ and $\mathsf{ConsFail}$.)

Lemma 1 (Margin and consistency). *Consider an execution (w, σ) with $w = w_1 \dots w_N$ and $\sigma = \sigma_1 \dots \sigma_{\mathrm{rnd}(N)}$. Let B be a block produced in slot $\ell \in [N]$, and let $t_0 > \ell$ be such that B is contained in some chain held by an honest party at time t_0. If for every $t \in \{t_0, \dots, N\}$ we have $\mu_\ell\left((w,\sigma)_{t\rceil}\right) < -B$ then B is contained in every chain C held by any honest party at any time $t \in \{t_0, \dots, N\}$.*

As a consequence, let $t^* > \ell$ be the earliest time such that B is contained in prune(C) *for some chain C held by an honest party P at time t^* (i.e., P considers B settled at time t^*). If for every $t \in \{t^*, \dots, N\}$ we have $\mu_\ell((w,\sigma)_{t\rceil}) < -B$ then B will not induce the event $\mathsf{ConsFail}$ during the execution.*

Proof. Let $F \vdash (w, \sigma)$ be the tree corresponding to the execution, and let T be a chain in $F_{t_0\rceil}$ that contains B and is dominant in $F_{t_0\rceil}$: such T exists by assumption, as $F_{t_0\rceil}$ describes the execution up to time t_0 and honest parties only hold chains that are dominant.

The proof proceed by induction on $t \in [t_0, N]$. For the base case, we first prove that at time t_0, no honest party is holding a chain that does not contain B. Assume the opposite, namely that an honest party is holding a chain T' at time t_0 and T' does not contain B. The fact that T' is held by an honest party implies that T' is dominant in $F_{t_0\rceil}$ and hence $\alpha_{F_{t_0\rceil}}(T') \geq 0$. However, we also have $\alpha_{F_{t_0\rceil}}(T) \geq 0$ for the same reason, and moreover, since T' does not contain B, we have $T \not\sim_\ell T'$ by definition of $\not\sim_\ell$. This implies $\mu_\ell\left((w,\sigma)_{t_0\rceil}\right) \geq \mu_\ell(F_{t_0\rceil}) \geq 0$, contradicting the assumption of the lemma.

For the inductive step, assume that for some $t \in [t_0, N-1]$ all dominant chains in $F_{t\rceil}$ contain B, and our goal is to prove the same is true for $t+1$. We consider two cases. First, assume that $\mathsf{wt}(\overline{F_{t\rceil}}) = \mathsf{wt}(\overline{F_{t+1\rceil}})$. Then any dominant chain in $F_{t\rceil}$ is also dominant in $F_{t+1\rceil}$, and contains B. Therefore, there exist dominant chains in $F_{t+1\rceil}$ that contain B, and as above, the existence of a dominant chain not containing B would contradict the assumption that $\mu_\ell\left((w,\sigma)_{t+1\rceil}\right)$ is negative. It remains to consider the case $\mathsf{wt}(\overline{F_{t+1\rceil}}) \geq \mathsf{wt}(\overline{F_{t\rceil}}) + 1$, we argue that $F_{t+1\rceil}$ again cannot contain dominant chains that do not contain B. Towards a contradiction, let T^* be such a chain, and let $T^*_{t\rceil}$ be its restriction to $F_{t\rceil}$. Note that the

weight of T^* could grow by at most $B+1$ in slot $t+1$ (compared to $T^*_{t\rceil}$), by obtaining at most one certificate and one block. Therefore, and since T is dominant in $F_{t+1\rceil}$, we have

$$\begin{aligned}\mu_\ell\left((w,\sigma)_{t\rceil}\right) &\geq \mu_\ell(F_{t\rceil}) \geq \alpha_{F_{t\rceil}}(T_{t\rceil}) = \mathsf{wt}(T_{t\rceil}) - \mathsf{wt}(\overline{F_{t\rceil}}) \\ &\geq \left(\mathsf{wt}_{F_{t+1\rceil}}(T) - B - 1\right) - \left(\mathsf{wt}(\overline{F_{t+1\rceil}}) - 1\right) = \alpha_{F_{t+1\rceil}}(T) - B \geq -B\,,\end{aligned}$$

contradicting the assumption of the lemma as desired. □

Given the above connection between margin and consistency, in Sections A.5–A.7 we state and prove recurrences describing the evolution of margin (and the dependent quantity reach) during the protocol's execution.

A.5 Reach and Margin Recurrences during 1-Rounds

We start by studying the behavior of reach and margin during a period in the execution of the protocol that takes the optimistic path. Namely, we look at a sequence of rounds that lead to a successful and timely creation of certificates, as represented by a sequence of symbols '1' in the voting string.

Theorem 1 (Reach and margin during 1-rounds). *Fix $\ell \geq 1$. Let (w,σ) and $(wx,\sigma 1^r)$ be two executions such that x exactly spans $r \cdot U$ slots corresponding to $r \geq 1$ voting rounds. We have $\rho(\varepsilon,\varepsilon) = 0$ and*

$$\rho(wx,\sigma 1^r) \leq \max\left\{\rho(w,\sigma) - r\cdot B + \#_{[\mathsf{ha}]}(x),\quad \#_{[\mathsf{a}]}(x) + D + \Delta\right\}\,.$$

Moreover, $\mu_\ell(wx,\sigma 1^r) \leq \rho(wx,\sigma 1^r)$, and if $\ell < |w| - D$ then

$$\mu_\ell(wx,\sigma 1^r) \leq \rho(w,\sigma) - r\cdot B + \#_{[\mathsf{ha}]}(x)\,.$$

Bounding reach. We first establish a helper lemma that, informally speaking, lower-bounds the growth of the weight of the public subtree during 1-rounds.

Lemma 2 (Public tree growth). *Let (w,σ) and $(wx,\sigma 1^r)$ be two executions such that x spans $r\cdot U$ slots exactly corresponding to $r \geq 1$ voting rounds. Let $F' \vdash (wx,\sigma 1^r)$ be a protocol-respecting tree and let $F \sqsubseteq F'$ be a restriction of F' to (w,σ). Then we have*

$$\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F}) + r\cdot B\,.$$

Proof. For each $i \in \{0,1,\ldots,r\}$ let F_i denote the restriction of F' to $\mathrm{lastSlt}(|\sigma|+i)$. Notice that we have $F_0 = F$ and $F_r = F'$. Moreover, for each $i \in [r]$, let v_i denote the vertex in F_{i-1} that becomes certified in round $|\sigma|+i$, and let J_i denote the corresponding justification, so we have $\mathrm{jslot}(J_i) = \mathrm{lastSlt}(|\sigma|+i-1)+1$ and $\overline{F_{i-1}} \sqsubseteq J_i \sqsubseteq F_{i-1}$. Let T_i be the maximum-weight chain in J_i that contains v_i, as guaranteed by (P3).

First, notice that for each $i \in [r-1]$ we have

$$\mathsf{wt}_{J_i}(T_i) \overset{(a)}{\leq} \mathsf{wt}_{J_{i+1}}(T_i) - B \overset{(b)}{\leq} \mathsf{wt}_{J_{i+1}}(T_{i+1}) - B\,. \tag{3}$$

To justify inequality (a), note that both T_i and the certificate on v_i appear in $\overline{F_i}$, as they were known to some honest party at the time the certificate was created and hence publicly

known $\Delta < U$ slots later; formally, both the justification J_i and the certificate $|\sigma|+i \in \mathcal{C}(F_i)$ are propagated in F_i . Therefore, they also appear in J_{i+1}, and $\mathsf{wt}_{J_{i+1}}(T_i)$ is at least $\mathsf{wt}_{J_i}(T_i)$ increased by the additional boost B provided by the newly added certificate. Inequality (b) then follows since T_{i+1} is maximum-weight in J_{i+1} and hence it is at least as heavy as any chain in $\overline{F_{i+1}}$, and as argued above, T_i appears in $\overline{F_{i+1}}$.

To conclude the argument, we now have

$$\mathsf{wt}(\overline{F_0}) \overset{(c)}{\leq} \mathsf{wt}_{J_1}(T_1) \overset{(d)}{\leq} \mathsf{wt}_{J_r}(T_r) - (r-1)\cdot B \overset{(e)}{\leq} \mathsf{wt}(\overline{F_r}) - r \cdot B\,. \tag{4}$$

Here, inequality (c) holds since $\overline{F_0} \sqsubseteq J_1$ and T_1 is maximum-weight in J_1. Inequality (d) is an $(r-1)$-fold application of (3). Finally, inequality (e) again holds as T_r appears in $\overline{F_r}$, and its weight has been increased since J_r by the final newly added certificate. Put together, (4) establishes the lemma. □

We now employ Lemma 2 to establish an upper bound for reach during a period of 1-rounds.

Lemma 3 (Reach upper bound). *Let (w,σ) and $(wx,\sigma\tau)$ be two executions such that x and τ span $r\cdot U$ slots exactly corresponding to $r \geq 1$ voting rounds. If $\tau = 1^r$ then*

$$\rho(wx,\sigma\tau) \leq \max\left\{\rho(w,\sigma) - rB + \#_{[\mathsf{ha}]}(x),\quad \#_{[\mathsf{a}]}(x) + D + \Delta\right\}\,.$$

Proof. Let $F' \vdash (wx,\sigma\tau)$ be a witness tree for $\rho(\cdot)$ and let T' be a witness chain in F', i.e.,

$$\rho(wx,\sigma\tau) = \alpha_{F'}(T') = \mathsf{wt}_{F'}(T') - \mathsf{wt}(\overline{F'})\,.$$

Let $F \sqsubseteq F'$ be a restriction of F' to (w,σ), let T be a restriction of T' to F.

We now consider two separate cases, depending on whether T' contains any vertices that were certified in rounds corresponding to τ, i.e., whether

$$\exists v \in T' \colon \mathsf{l}_{\mathsf{c}}(v) \cap \{|\sigma|+1,\ldots,|\sigma\tau|\} \neq \emptyset\,. \tag{5}$$

Let us first consider the case that (5) is satisfied in F'. Let $i \in [r]$ be the largest index such that the certificate from round $|\sigma|+i$ certifies a block on T'. We have $\tau_i = 1$ and $\mathrm{lastSlt}(|\sigma|+i) \leq |wx|$, hence the certificate, and the block it certifies, appear in $\overline{F'}$ by axiom (C3) and the definition of a public subtree. This in turn implies

$$\alpha_{F'}(T') \leq \#_{[\mathsf{a}]}(x) + D + \Delta\,,$$

as T' may, on top of the deepest certified block on it (which appears in $\overline{F'}$), only contain

- at most D vertices from slots corresponding to the last round of σ (due to (P3));
- at most $\#_{[\mathsf{a}]}(x)$ adversarial vertices from slots corresponding to x; and
- at most Δ honest vertices from x that do not appear in $\overline{F'}$.

This concludes the proof of the first case.

Now consider the case where (5) is not satisfied in F', i.e., there is no certificate associated with a round described by τ and certifying any of the vertices on T'. Based on that, we have

$$\mathsf{wt}_{F'}(T') \leq \mathsf{wt}_F(T) + \#_{[\mathsf{ha}]}(x)\,, \tag{6}$$

as the weight of T' grows on top of the weight of T only due to any additional vertices in $T' \setminus T$. On the other hand, we can lower-bound $\mathsf{wt}(\overline{F'}) - \mathsf{wt}(\overline{F})$ using Lemma 2, getting

$$\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F}) + rB \,. \tag{7}$$

Inequalities (6) and (7) then together imply

$$\rho(wx, \sigma\tau) \leq \rho(w, \sigma) - rB + \#_{[\mathsf{ha}]}(x) \,,$$

concluding the proof for this case as well. □

Bounding margin. Towards bounding the quantity $\mu_\ell(\cdot)$, first observe that its definition directly implies that $\mu_\ell(w, \sigma) \leq \rho(w, \sigma)$ for any execution (w, σ). Moreover, for any (w, σ) with $|w| < \ell$, we actually have $\mu_\ell(w, \sigma) = \rho(w, \sigma)$ as, recalling the definition of $\mu_\ell(F)$ and the relation $\not\sim_\ell$, notice that any chain T with $\mathsf{l}_\#(T) < \ell$ satisfies $T \not\sim_\ell T$, and hence the witness chains T_1, T_2 for $\mu_\ell(F)$ may satisfy $T_1 = T_2$.

We now proceed to prove an upper bound on μ_ℓ.

Lemma 4 (Margin upper bound). *Let (w, σ) and $(wx, \sigma 1^r)$ be two executions such that x spans $r \cdot U$ slots exactly corresponding to $r \geq 1$ voting rounds. Fix $\ell < |w| - D$. Then*

$$\mu_\ell(wx, \sigma 1^r) \leq \rho(w, \sigma) - rB + \#_{[\mathsf{ha}]}(x) \,.$$

Proof. Let $F' \vdash (wx, \sigma\tau)$ be a witness tree for $\mu_\ell(\cdot)$ and let $T'_1 \not\sim_\ell T'_2$ be a pair of witness chains in F' such that $\alpha_{F'}(T'_1) \geq 0$ and $\alpha_{F'}(T'_1) \geq \alpha_{F'}(T'_2) = \mu_\ell(wx, \sigma 1^r)$. Let $F \sqsubseteq F'$ be a restriction of F' to (w, σ). Let T_1, T_2 be restrictions of T'_1, T'_2 to F, respectively. Observe that Lemma 2 gives us $\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F}) + rB$.

Let $\mathcal{R} = \{|\sigma| + 1, \ldots, \sigma + r\}$ denote the indices of the final r 1-rounds in $(wx, \sigma 1^r)$. By axiom (P4) we know that all certificates corresponding to rounds in $\mathcal{R}$ appear on the same chain, and by axiom (P3), each of them certifies a block belonging to a slot from round $|w| - D$ or later. Hence $T'_1 \not\sim_\ell T'_2$ together with the assumption $\ell < |w| - D$ implies that at least one of the chains T'_1, T'_2 contains no vertices certified in these rounds. We now consider these two cases separately.

If T'_2 contains no certificates from rounds in $\mathcal{R}$ then we immediately have

$$\mathsf{wt}_{F'}(T'_2) \leq \mathsf{wt}(F) + \#_{[\mathsf{ha}]}(x) \,, \tag{8}$$

as the weight of T_2 only grows during the rounds in $\mathcal{R}$ by added vertices, and there are at most $\#_{[\mathsf{ha}]}(x)$ of them. Combining Lemma 2, (8), and the fact that $\rho(w, \sigma) \geq \rho(F) = \mathsf{wt}(F) - \mathsf{wt}(\overline{F})$ for any blocktree $F \vdash (w, \sigma)$, we get

$$\begin{aligned} \mu_\ell(wx, \sigma 1^r) = \alpha_{F'}(T'_2) &= \mathsf{wt}_{F'}(T'_2) - \mathsf{wt}(\overline{F'}) \\ &\leq \big(\mathsf{wt}(F) + \#_{[\mathsf{ha}]}(x)\big) - \big(\mathsf{wt}(\overline{F}) + rB\big) \\ &\leq \rho(w, \sigma) - rB + \#_{[\mathsf{ha}]}(x) \,, \end{aligned}$$

concluding the proof for this case.

On the other hand, if T'_1 contains no certificates from rounds in $\mathcal{R}$ then

$$\begin{aligned} \mathsf{wt}_{F'}(T'_1) &\leq \mathsf{wt}(F) + \#_{[\mathsf{ha}]}(x) \leq \mathsf{wt}(\overline{F}) + \rho(F) + \#_{[\mathsf{ha}]}(x) \\ &\leq \mathsf{wt}(\overline{F}) + \rho(w, \sigma) + \#_{[\mathsf{ha}]}(x) \end{aligned} \tag{9}$$

and, again using Lemma 2,

$$\mathsf{wt}_{F'}(T_2') = \mu_\ell(F') + \mathsf{wt}(\overline{F'}) \geq \mu_\ell(F') + \mathsf{wt}(\overline{F}) + rB = \mu_\ell(wx, \sigma 1^r) + \mathsf{wt}(\overline{F}) + rB\,. \quad (10)$$

Combining (9) and (10) and considering that by choice of T_1', T_2' we have $\mathsf{wt}_{F'}(T_1') \geq \mathsf{wt}_{F'}(T_2')$ concludes the proof also for the second case. □

A.6 Reach and Margin Recurrences during 0-Rounds

We now turn our attention towards rounds that do not lead to timely certificates, and are represented by '0'-symbols in the voting string. This corresponds to the cooldown periods, and intuitively, in these periods reach and margin behave analogously to an execution of a plain longest-chain PoS protocol without any certificates. This behavior has already been studied in previous work [24] on which our analysis for this case relies. We will use the following terminology from [24,22].

Definition 15 (Terminal leader strings; phases). *Let $\Sigma_{\mathsf{A}} = (\{0\} \times \mathbb{N}) \subset \Sigma$. A leader string w is called* terminal *if it is either the empty string or it terminates with a Δ-period with no honest successes, i.e., if $w \in \{\varepsilon\} \cup \left(\Sigma^* \circ \Sigma_{\mathsf{A}}^{\Delta}\right)$, where $\circ$ denotes language concatenation. A non-empty terminal leader string ϕ is called a* phase *if it ends with the first string from $\Sigma_{\mathsf{A}}^{\Delta}$ it contains. Formally, $\phi = \phi_1 \dots \phi_n \in \Sigma^n$ with $n \geq \Delta$ is a phase if $(\phi_{i-\Delta+1} \dots \phi_i \in \Sigma_{\mathsf{A}}^{\Delta}) \Rightarrow i = n$.*

We remark that any characteristic string $w \in \Sigma^n$ has a unique decomposition into phases in that sense that w can be written $\phi^{(1)} \cdots \phi^{(s)}\psi$, where each $\phi^{(i)}$ is a phase and ψ is a string that does not contain a sequence from $\Sigma_{\mathsf{A}}^{\Delta}$. We call ψ an "incomplete phase" and note that ψ may be empty, in which case the string w is terminal.

Moreover, recall that the length of an execution in slots does not need to be an integer multiple of U.

Theorem 2 (Reach and margin during 0-rounds). *Fix $\ell \geq 1$. Let (w, σ) be an execution and (wx, σ') be its extension such that all slots covered by x correspond to 0-rounds in σ', i.e., if we write $\sigma' =: \sigma'_1 \dots \sigma'_{|\sigma'|} \in \{0, ?, 1\}^{|\sigma'|}$ then $\forall i \in \{|w|+1, \dots, |wx|\}\colon \sigma'_{\mathrm{rnd}(i)} = 0$. We have*

$$\mu_\ell(wx, \sigma') \leq \rho(wx, \sigma') \leq \rho(w, \sigma) + \#_{[\mathsf{ha}]}(x)\,. \quad (11)$$

Moreover, if w is terminal and $x =: \phi$ is a phase, we have

$$\rho(w\phi, \sigma') \leq \max\left\{\rho(w, \sigma) + \#_{[\mathsf{a}]}(\phi) - h_\Delta(\phi), \#_{[\mathsf{a}]}(\phi)\right\}$$

and if additionally $\mu_\ell(w, \sigma) < -\#_{[\mathsf{a}]}(\phi)$, we also have

$$\mu_\ell(w\phi, \sigma') \leq \mu_\ell(w, \sigma) + \#_{[\mathsf{a}]}(\phi) - h_\Delta(\phi)\,.$$

Finally, if w is terminal, $\phi = (1,0)(0,0)^\Delta$, and $\rho(w, \sigma) = \mu_\ell(w, \sigma) = 0$ then $\mu_\ell(w\phi, \sigma') \leq -1$.

Proof. The first inequality in (11) follows directly from the definitions. For the second inequality, let $F' \vdash (wx, \sigma')$ be a witness tree for $\rho(\cdot)$ and let T' be a (reach) witness chain in F', i.e., $\rho(wx, \sigma') = \alpha_{F'}(T') = \mathsf{wt}_{F'}(T') - \mathsf{wt}(\overline{F'})$. Let $F \sqsubseteq F'$ be a restriction of F' to (w, σ); let T be a restriction of T' to F. We have $\mathsf{wt}_{F'}(T') \leq \mathsf{wt}_F(T) + \#_{[\mathsf{ha}]}(x)$as T'

might grow—compared to T—by at most $\#_{[\mathsf{ha}]}(x)$ vertices, and it contains no additional certificates compared to T. On the other hand, it follows directly from the definition of a public subtree that $\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F})$. Combining these two observations, we get

$$\begin{aligned}\rho(wx, \sigma') &= \mathsf{wt}_{F'}(T') - \mathsf{wt}(\overline{F'}) \leq (\mathsf{wt}_F(T) + \#_{[\mathsf{ha}]}(x)) - \mathsf{wt}(\overline{F}) \leq \rho(F) + \#_{[\mathsf{ha}]}(x) \\ &\leq \rho(w, \sigma) + \#_{[\mathsf{ha}]}(x)\end{aligned}$$

as desired.

The remaining claims of the theorem are direct reformulations of Theorem 2 from [24] to our setting. Indeed, [24, Theorem 2] describes the behavior of reach and margin in the PoS setting without the presence of certificates, which is exactly the setting that occurs in our analysis during 0-rounds. To see the connection, notice that our blocktree axioms (T1), (T2), (C1) and (P2) correspond to blocktree axioms (A1), (S3), (S4) and (A2) in [24], respectively; while our axioms (T3), (C2), (C3), (P3) and (P4) govern the behavior of certificates and have no counterparts in [24]. Furthermore, an inspection of the definitions of reach and margin shows that in the absence of certificates, both notions coincide with their counterparts in [24], justifying the translation of their results to our 0-rounds. □

A.7 Reach and Margin Recurrences during ?-Rounds

Finally, we study the effect of '?'-rounds. Intuitively, these rounds are detrimental to the evolution of reach and margin, but the following theorem upper-bounds the loss from a '?'-round by the term $B + U$, and as we will see later, this is sufficient for our analysis as '?'-rounds are sufficiently rare.

Theorem 3 (Reach and margin during ?-rounds). *Fix $\ell \geq 1$. Let (w, σ) and $(wx, \sigma?)$ be two executions such that x exactly spans U slots corresponding to a single ?-round. Then*

$$\rho(wx, \sigma?) \leq \rho(w, \sigma) + B + U \qquad \text{and} \qquad \mu_\ell(wx, \sigma?) \leq \mu_\ell(w, \sigma) + B + U\,.$$

Proof. Reach. Let $F' \vdash (wx, \sigma?)$ be a witness tree for $\rho(\cdot)$ and let T' be a (reach) witness chain in F', i.e., $\rho(wx, \sigma?) = \alpha_{F'}(T') = \mathsf{wt}_{F'}(T') - \mathsf{wt}(\overline{F'})$. Let $F \sqsubseteq F'$ be a restriction of F' to (w, σ); let T be a restriction of T' to F. We have

$$\mathsf{wt}_{F'}(T') \leq \mathsf{wt}_F(T) + B + U$$

as T' might grow—compared to T—by at most U vertices, and also contain at most one additional certificate (from the final ?-round), contributing with weights at most U and B, respectively. On the other hand, it follows directly from the definition of a public subtree that $\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F})$. Combining these two observations, we get

$$\begin{aligned}\rho(wx, \sigma?) &= \mathsf{wt}_{F'}(T') - \mathsf{wt}(\overline{F'}) \leq (\mathsf{wt}_F(T) + B + U) - \mathsf{wt}(\overline{F}) \leq \rho(F) + B + U \\ &\leq \rho(w, \sigma) + B + U\end{aligned}$$

as desired.

Margin. Let now $F' \vdash (wx, \sigma?)$ denote a witness tree for $\mu_\ell(\cdot)$ and let $T'_1 \not\sim_\ell T'_2$ be a pair of witness chains in F' such that $\alpha_{F'}(T'_1) \geq 0$ and $\alpha_{F'}(T'_1) \geq \alpha_{F'}(T'_2) = \mu_\ell(wx, \sigma?)$. Let $F \sqsubseteq F'$ be a restriction of F' to (w, σ). Let T_1 and T_2 be restrictions of T'_1 and T'_2 to F, respectively; notice that $T_1 \not\sim_\ell T_2$. By the same argument as in the case of reach above, any

chain T' in F' satisfies $\mathsf{wt}_{F'}(T') \leq \mathsf{wt}_F(T) + B + U$, where T is the restriction of T' to F. We also have $\mathsf{wt}(\overline{F'}) \geq \mathsf{wt}(\overline{F})$. Therefore

$$\begin{aligned}\mu_\ell(w,\sigma) \geq \mu_\ell(F) &= \max_{\substack{T_a,\ T_b \text{ chains in } F \\ T_a \not\sim_\ell T_b}} \Big(\min\{\mathsf{wt}_F(T_a), \mathsf{wt}_F(T_b)\}\Big) - \mathsf{wt}(\overline{F}) \\ &\geq \min\{\mathsf{wt}_F(T_1), \mathsf{wt}_F(T_2)\} - \mathsf{wt}(\overline{F}) \\ &\geq (\mathsf{wt}_{F'}(T_2') - U - B) - \mathsf{wt}(\overline{F'}) = \mu_\ell(wx, \sigma?) - U - B\end{aligned}$$

as desired. □

A.8 Good Leader Strings

Having established recurrences for reach and margin given a particular execution (w, σ), the next step in the proof is to establish that the voting string σ always has a certain format; this is the purpose of Section A.9. In preparation, this section introduces the notion of *good leader strings* w, which necessitates defining two notions that have close correspondence with reach and margin (cf. Section A.3).

Recall that *reach* is a quantity that intuitively captures weight stemming from blocks and certificates unknown to honest parties, and that *margin relative to some slot* ℓ corresponds to the maximum weight difference between two dominant chains that at slot ℓ or before.

In the following, let $\Phi = (\phi^{(1)}, \ldots, \phi^{(k)})$ be a sequence of phases in Σ^*.[10]

Reach-like quantity. Consider a quantity $R[\Phi; z]$ defined as follows, where $z > 0$ is an initial value and $R[\Phi; z]$ corresponds to the reach after processing Φ in a setting without certificates:

$$R[\Phi; z] \ := \ \begin{cases} z & \text{if } k = 0 \\ \max\big(R[\Phi'; z]) + \#_{[\mathsf{a}]}\phi^{(k)} - h_\Delta \phi^{(k)}, \#_{[\mathsf{a}]}\phi^{(k)}\big) & \text{otherwise,} \end{cases}$$

where $\Phi' = (\phi^{(1)}, \ldots, \phi^{(k-1)})$. Observe the correspondence between the evolution of R and that of ρ during 0-rounds (cf. Section A.6).

Margin-like quantity. In a similar vein, consider a quantity $M[\Phi; z]$ defined as follows: intuitively, $z > 0$ represents an initial value of both reach and margin, and $M[\Phi; z]$ corresponds to margin μ_ℓ with respect to slot ℓ, starting right after ℓ and processing Φ, in a setting without certificates. Hence, M behaves as follows (for Φ' as above): if $R[\Phi'; z] = M[\Phi'; z] = 0$ and $\phi^{(k)} = (1,0)(0,0)^\Delta$, then

$$M[\Phi; z] \ := \ -1\,;$$

otherwise,

$$M[\Phi; z] \ := \ \begin{cases} M[\Phi'; z] + \#_{[\mathsf{a}]}\phi^{(k)} - h_\Delta \phi^{(k)} & \text{if } M[\Phi'; z] < -\#_{[\mathsf{a}]}\phi^{(k)}, \\ R[\Phi; z] & \text{otherwise.} \end{cases}$$

Observe the correspondence between the evolution of M and that of μ_ℓ during 0-rounds (cf. Section A.6).

[10] Consult Section A.1 for notation related to lotteries, such as Σ, and Section A.6 for the definition of a phase.

Leader-string properties. Consider the following properties of a leader-string:

Definition 16 (Reach- and margin-like LS properties). *A leader string w has*

- $(T;\alpha,\beta)$-reach *if for every contiguous substring x of w of length at least T,*

$$R[\phi^{(1)}\cdots\phi^{(t)};\alpha+\#_{[\mathsf{ha}]}\phi^{(0)}]+\#_{[\mathsf{ha}]}\psi\le\beta\,,$$

and

- $(T;\alpha,\beta)$-(super-)margin *if for every contiguous substring x of w of length at least T,*

$$M_\ell[\phi^{(1)}\cdots\phi^{(t)};\alpha+\#_{[\mathsf{ha}]}\phi^{(0)}]+\#_{[\mathsf{ha}]}\psi\le\beta$$

(and if additionally "margin remains negative at all times"),

where $x=\phi^{(0)}\cdots\phi^{(t)}\psi$ is the phase decomposition of x into phases where ψ is a (perhaps empty) incomplete phase with no quiet period.

Definition 17 (Chain-quality-like LS property for cooldown). *A leader-string w has* (T_{HCI},B,D,U)-cooldown-chain-quality *if any substring*

$$z\;=\;x_{\mathsf{pre}}\|\underbrace{x_1\|\ldots\|x_t}_{x}\|\underbrace{y_{\mathsf{pre}}\|y_{\mathsf{hci}}\|y_{\mathsf{post}}}_{y}$$

of w satisfying

1. *y_{hci} is round-aligned (i.e., starting at round boundary),*
2. *$|y_{\mathsf{hci}}|=(T_{\mathsf{HCI}}-3)\cdot U$,*
3. *$|y_{\mathsf{post}}|\ge 0$,*
4. *if $|y_{\mathsf{pre}}|\le 3U$, then $t=0$, else $t\ge 0$, and*
5. *$|x_{\mathsf{pre}}|\ge 0$,*
6. *$|x_i|=U$,*

has the property that

$$\begin{aligned}h_\Delta(\tilde{x}_{\mathsf{pre}})+\sum_{i=1}^{t}h_\Delta(\tilde{x}_i)+h_\Delta(\tilde{y})\;&>\;\#_{[\mathsf{a}]}(x_{\mathsf{pre}})+\sum_{i=1}^{t}\#_{[\mathsf{a}]}(x_i)+\#_{[\mathsf{a}]}(y)+(B+D+U+\Delta)\\&=\;\#_{[\mathsf{a}]}(z)+(B+D+U+\Delta)\,,\end{aligned}\tag{12}$$

where

- *$\tilde{y}$ (resp. $\tilde{x}_{\mathsf{pre}}$) is y (resp. x_{pre}) with Δ symbols truncated on each side,*
- *$\tilde{x}_i$ is x_i with 2Δ symbols truncated on the left and Δ on the right.*

Definition 18 (Chain-quality-like LS property for happy periods). *A leader-string w has* (T_{HL},D,U)-happy-chain-quality *if any substring*

$$z\;=\;x_{\mathsf{pre}}\|\underbrace{x_1\|\ldots\|x_t}_{x}$$

of w satisfying

1. *x_1 is round-aligned (i.e., starting at round boundary),*

2. $0 \leq |x_{\mathsf{pre}}| < U$,
3. $|x_i| = U$,
4. $t \geq T_{\mathsf{HL}}$

has the property that

$$h_{\Delta}(\tilde{x}_{\mathsf{pre}}) + \sum_{i=1}^{t} h_{\Delta}(\tilde{x}_i) \;>\; \#_{[\mathsf{a}]}(x_{\mathsf{pre}}) + \sum_{i=1}^{t} \#_{[\mathsf{a}]}(x_i) + (D + U + \Delta) \qquad (13)$$
$$= \#_{[\mathsf{a}]}(z) + (D + U + \Delta)\ ,$$

where

- $\tilde{x}_{\mathsf{pre}}$ *is* x_{pre} *with* Δ *symbols truncated on each side,*
- $\tilde{x}_i$ *is* x_i *with* 2Δ *symbols truncated on the left and* Δ *on the right.*

Good leader strings. The following definition captures the exact requirements a leader string must satisfy for the protocol to be secure:

Definition 19 (Good leader strings). *A leader string* $w \in \Sigma^*$ *is called* $(T_{\mathsf{HCI}}, T_{\mathsf{CS}}, T_{\mathsf{HL}}, B, U, L, \kappa_1, \kappa_2)$-good *if it has*

- $(T_{\mathsf{HCI}} \cdot U; \kappa_1 + B + U, \kappa_1)$*-reach;*
- $(T_{\mathsf{CS}} \cdot U; \kappa_1, -\kappa_2)$*-margin;*
- $(T_{\mathsf{CS}} \cdot U; -\kappa_2 + 2L + B + U, -\kappa_2)$*- and* $(T \cdot U; -\kappa_2 + 2L + B + U, -1)$*-super-margin for every* $T \geq 1$*;*
- $(T_{\mathsf{HCI}}, B, D, U)$*-cooldown-chain-quality;*
- (T_{HL}, D, U)*-happy-chain-quality.*

Otherwise, w *is called* $(T_{\mathsf{HCI}}, T_{\mathsf{CS}}, T_{\mathsf{HL}}, B, U, L, \kappa_1, \kappa_2)$-bad.

For simplicity, whenever the parameters are clear from the context, they are dropped, and we simply say *good leader string.*

Lemma 5. *Let* $\kappa \in \mathbb{N}$ *be the security parameter, and let* $w = w_1 \dots w_N \in \Sigma^N$ *for* $N \in \mathsf{poly}(\kappa)$ *be a leader string sampled from* $\mathcal{D}(r_{\mathsf{h}}, r_{\mathsf{a}})$*. For any* $\kappa_1, \kappa_2 \in \Theta(\kappa)$*,* $U, L \in O(1)$*, and* $B \in O(\kappa)$ *such that* $B \leq \kappa_2/2$*, there exist* $T_{\mathsf{HCI}}, T_{\mathsf{CS}}, T_{\mathsf{HL}} \in O(\kappa)$ *such that*

$$\Pr\left[w \text{ is } (T_{\mathsf{HCI}}, T_{\mathsf{CS}}, T_{\mathsf{HL}}, B, U, L, \kappa_1, \kappa_2)\text{-bad}\right] \leq \mathit{negl}(\kappa)\ .$$

Proof (sketch). It suffices to prove that each of the properties required for $(T_{\mathsf{HCI}}, T_{\mathsf{CS}}, T_{\mathsf{HL}}, B, U, L, \kappa_1, \kappa_2)$-goodness in Definition 19 is violated with probability negligible in κ; the lemma then follows by a union bound.

Reach. Under our honest-majority assumption (2), the recursion defining $R[\cdot;\cdot]$ implies that, over any substring of w, R evolves essentially as a negatively biased random walk with a barrier at 0. An analogous walk is analyzed in [20, Sec. 4.1]; in particular, given that $B + U \in O(\kappa)$, one can choose $T_{\mathsf{HCI}} \in O(\kappa)$ so that over T_{HCI} rounds it descends by $B + U$ to end below κ_1 with overwhelming probability.

Margin. The reasoning is similar, slightly complicated by the dependence of $M[\cdot;\cdot]$ on $R[\cdot;\cdot]$ (notice that both quantities start with the same value κ_1 at the beginning of the observed walk). The quantity $M[\cdot;\cdot]$ evolves as a negatively biased random walk but exhibits

special behavior around 0 where it is affected by the current value of $R[\cdot;\cdot]$. This type of walk is studied in [20, Sec. 4.2], and one can conclude that $T_{\mathsf{CS}} \in O(\kappa)$ can be chosen so that it descends from an initial value κ_1 below $-\kappa_2$ with overwhelming probability.

Super-margin. Here the situation is simpler in the sense that $M[\cdot;\cdot]$ already starts at a large negative value $-\kappa_2 + 2L + B + U \in -\Omega(\kappa)$ (here we leverage $B \leq \kappa_2$). As long as it remains sufficiently below 0, its evolution is a negatively biased random walk that is unaffected by the behavior of $R[\cdot;\cdot]$. As such, the probability that it ever climbs to zero is negligible in κ, and one can choose $T_{\mathsf{CS}} \in O(\kappa)$ so that over a T_{CS}-round long interval it descends by $2L + B + U$ with overwhelming probability.

Chain quality. The arguments for chain quality during cooldowns and happy phases are similar. For an appropriate choice of $U = O(1)$ satisfying

$$U > \frac{3\Delta}{1 - p_a\left(\Delta + \frac{1}{p_h}\right)}$$

the expected honest growth h_Δ in a single round, even after removing a Δ-dependent prefix and suffix, dominates the expected number of adversarial slots in the whole round. The desired statements then follow for sufficiently long sequences of rounds (i.e., for $T_{\mathsf{HCI}}, T_{\mathsf{HL}} = \Omega(\kappa)$) via a Chernoff-style bound. □

A.9 Voting String Analysis

The goal of this section is to prove that the voting string always obeys the rules provided in Theorem 4. Throughout the entire section, whenever there is mention of a good leader string, what is in fact meant is a $(T_{\mathsf{HCI}}, T_{\mathsf{CS}}, B, U, L, \kappa_1, \kappa_2)$-*good* leader string, where κ_1 and κ_2 are two security parameters.

Theorem 4. *Assume a good leader string is chosen. Then, the voting string is built according to the following rules (where λ is the empty string and $\longrightarrow$ stands for "can be followed by"):*

(HS-I)	$\sigma = \lambda$	$\longrightarrow$	1		
(HS-II)	$\sigma = \ldots 1$	$\longrightarrow$	$\{?, 1\}$		
(HS-III)	$\sigma = \ldots ?$	$\longrightarrow$	0		
(HS-IV)	$\sigma = \ldots 1\ ?\ 0^L$	$\longrightarrow$	0	*if*	$1 \leq L < K - 1$
(HS-V)	$\sigma = \ldots 1\ ?\ 0^L$	$\longrightarrow$	$\{?, 1\}$	*if*	$L \in \{K-2, K-1\}$
(HS-VI)	$\sigma = \ldots 0\ ?\ 0^L$	$\longrightarrow$	0	*if*	$1 \leq L < K - 1$
(HS-VII)	$\sigma = \ldots 0\ ?\ 0^L$	$\longrightarrow$	$\{?, 1\}$	*if*	$L = K - 1$

A voting string $\sigma \in \{0, 1, ?\}^*$ is called *valid* if it satisfies the conditions imposed by Theorem 4. Note that a voting string can naturally be split into cycles $\sigma = c_1 c_2 \ldots c_\ell$ such that

$$c_i \ = \ 1^t\ ?\ 0^L$$

for $t \geq 0$ (except for c_1, where $t \geq 1$) and $L \in \{K-1, K-2\}$; note that c_ℓ may be a partial cycle. Further, each cycle c can be subdivided into periods as follows (where the partial cycle at the end may only have some of the periods):

- *happy period*: the (maximal) prefix of 1's in c;
- *?-period*: the ?-round following the happy period together with the two subsequent 0-rounds;

– *healing/certificate-inclusion (HCI) period*: the sequence of T_{HCI} 0-rounds following a ?-period;
– *certificate-settlement (CS) period*: the sequence of T_{CS} 0-rounds following the HCI period;
– *leftover period*: the remaining 0-rounds in the cycle.

Note that the reason for assigning the first two 0-rounds after the ?-round to the ?-period is that honest parties can only conclusively determine that the protocol is in cooldown after not seeing a certificate during two complete rounds, which is required for them to submit the latest certificate they know to the chain (the CI in HCI). Therefore, it is convenient to define the HCI period as the period in which there must be an honest block, even though healing technically starts earlier.

Helper lemmas. The proof of Theorem 4 is by induction. To facilitate this, one formalizes the helper lemmas below.

Cooldown properties. The first lemma establishes properties of the cooldown portion of cycles (HCI and CS periods).

Lemma 6. *Assume a good leader string is chosen and that the execution has yielded a valid voting string $\sigma \in \{0,1,?\}^M$ up to some round M. Let $\mathcal{D}_M$ be the set of chains that are dominant at the end of round M.*[11] *Then,*

1. *all chains in $\mathcal{D}_M$ agree up to the end of the last HCI period that is followed by a complete CS period.*
2. *every chain in $\mathcal{D}_M$ has at least one honest block in every complete HCI period, and*

Note that proof of the second part of Lemma 6 uses the first part.

Agreement on HCI periods. The proof of the first part proceeds by first tracking reach ρ from genesis, showing that it remains bounded throughout, and in particular below security parameter κ_1 after the last slot ℓ of the HCI period in question. Then, μ_ℓ, i.e., margin relative to ℓ is tracked, starting at κ_1 after slot ℓ, showing that it falls below $-\kappa_2$ by the end of the subsequent CS period and then that it remains negative until the end of round M, which yields the desired claim in conjunction with Lemma 1. For readability, the arguments of reach and margin are dropped in favor of simply thinking them as a function of time.

Observe that at genesis ρ is 0. This is followed by zero or more 1-rounds, after which, by virtue of Theorem 1 and the assumption that $B \geq U$, ρ is bounded by $U + 2L + \Delta \leq \kappa_1$ (the last inequality by assumption again). Similarly, by Theorem 3, ρ is bounded by $\kappa_1 + B + U$ after the subsequent ?-round.

The goal is now to apply the good-leader-string (GLS) property to argue that ρ is bounded by κ_1 either (a) after the final round before the subsequent restart or (b) after the final round of the subsequent HCI period if that period is already the HCI period in question.

(a) In this case, let x be the leader string corresponding to the entire period of 0-rounds. Decompose it into $x = \phi^{(0)} \dots \phi^{(k)} \psi$ be the decomposition of x into phases where ψ is a (perhaps empty) incomplete phase with no quiet period. Note that reach is at most $\kappa_1 + B + U + \#_{[\mathsf{ha}]}\phi^{(0)}$ after phase $\phi^{(0)}$. Thus, by the GLS property and the fact that

[11] A chain is *dominant at time t* if it sufficiently heavy to be adopted by an honest party.

$|x| \geq U \cdot T_{\mathsf{HCI}}$, reach is at most $\kappa_1 - \#_{[\mathsf{ha}]}\psi$ after $\phi^{(k)}$ and at most κ_1 after the entire period of 0-rounds.

(b) A similar argument, but taking as x only the leader string up to the end of that last round of the HCI in question.

In case (a), observe that after the zero or more 1-rounds following the cooldown, ρ remains below κ_1, again using Theorem 1 as well as $B \geq U$ and $U + 2L + \Delta \leq \kappa_1$. The tracking now continues as above until the HCI period in question.

One now needs to track μ_ℓ, starting at ℓ, where it is bounded by reach, i.e., at that point, $\mu_\ell \leq \rho \leq \kappa_1$. Analogously to case (a) above, one considers leader string portion corresponding to the remaining cooldown, i.e., to the at least T_{CS} 0-rounds following the HCI period in question, and argues that, based on the GLS property, μ_ℓ drops to below $-\kappa_2$.

It remains to show that μ_ℓ remains negative until round M. After the zero or more 1-rounds that follow, $\mu_\ell \leq -\kappa_2 + 2L$, using Theorem 1 and $B \geq U$. After the subsequent ?-round, $\mu_\ell \leq -\kappa_2 + 2L + B + U$. It is also clear that during these rounds μ_ℓ cannot become non-negative.

If M has not been reached yet, consider again two cases (a) and (b), depending on whether round M lies beyond the next cooldown or not, respectively.

(a) In the above fashion and using the first super-margin GLS property, argue that $\mu_\ell \leq -\kappa_2$ by the end of the cooldown and continue tracking margin in this fashion.

(b) In the above fashion and using the second super-margin GLS property, argue that $\mu_\ell \leq -1$ by the end of round M.

This concludes the proof of the first part of Lemma 6.

Honest blocks in HCI periods. The proof of the second part of Lemma 6 proceeds by induction on the number of cycles, using the first part of the lemma to argue settlement of the purported honest block in each HCI period by the end of the subsequent CS period.

Consider the first cycle c_1 and a slot s anywhere in c_1 but after the HCI period. Assume, towards a contradiction, that there is a chain C dominant in slot s without any honest blocks with labels from the HCI period in c_1. Let s_{left} be the slot number of the last honest block B^* before the HCI period; note that B^* may be the genesis block. Further, let s_{right} be the first slot after the HCI period for which $C[:s_{\mathsf{right}}]$ is viable; this happens no later than at slot s. Observe that there are no honest blocks in the interval $(s_{\mathsf{left}}, s_{\mathsf{right}})$ on C.

One now (I) lower bounds the minimum honest weight (MHW)—the minimum weight among all honestly held chains—at slot s_{right} and (II) argues that C cannot reach this MHW by slot s_{right} (based on the assumption that a good leader string was chosen). In the following, let $w - 1$ be the weight of C in slot $s_{\mathsf{left}} - 1$—as seen by the honest party P^* who created B^*.

Let r_{left} be the round that contains $s_{\mathsf{left}} + \Delta$. Consider the following two cases:

(A) r_{left} is a 1-round;

(B) r_{left} is a ?-round or a 0-round.

In case (A), let $r_{\mathsf{left}} + 1, \ldots, r_{\mathsf{left}} + t$ be the $t \geq 0$ 1-rounds following r_{left} and let $x_1, \ldots, x_t$ be the leader string for those portions. Further, let y_{pre}, y_{hci}, and y_{post} denote the leader-string during the subsequent ?-period, HCI period, and the slots between the end of the HCI period and s_{right}. Note that

$$z := \underbrace{x_1 \| \ldots \| x_t}_{x} \| \underbrace{y_{\mathsf{pre}} \| y_{\mathsf{hci}} \| y_{\mathsf{post}}}_{y}$$

satisfies the five conditions of Definition 17.

Towards (I), the first step is to show:

Proposition 1. *The MHW is at least $w - C + B$ at the end of round r_{left}.*

Proof. Consider first the case where s_{left} is later than 2Δ after the beginning of r_{left}. In this case, $w-1$ includes the weight of the certificate from r_{left}, as it is known to all honest parties 2Δ into r_{left}.

In the other case, note that is suffices to show that the MHW at the beginning of r_{left} is at least $w - C$ as then the fact that r_{left} is a 1-round implies that the MHW grows to at least $w + B$ by the end of r_{left}.

If the certificate form r_{left} ends up on C, then C has weight $w + B \geq w - C + B$ by the end of r_{left}. Otherwise, let $i \geq 1$ be such that the certificate from $r_{\mathsf{left}} - i$ is the last certificate on C (this may go all the way back to the genesis certificate). Let C' be a chain that is dominant at the beginning of r_{left} and w' its weight; note that the MHW at that point is thus at least w'. It remains to show that $w' \geq w - C$.

Let w'' be the weight of the chain ending at the block B'' certified in round $r_{\mathsf{left}} - i$ and observe that

- $w' \geq w'' + iB$ as C' contains all the certificates from rounds $r_{\mathsf{left}} - (i-1), \ldots, r_{\mathsf{left}}$, and
- $w - w'' \leq C + iU$ as this is the maximum number of blocks that C could gain between B'' and B^*.

Using that $B \geq U$, the above inequalities imply that $w' \geq w - C$. □

After the $t \geq 0$ 1-rounds $r_{\mathsf{left}} + 1, \ldots, r_{\mathsf{left}} + t$, the MHW is at least

$$w - C + tB + \sum_{i=1}^{t} h_{\Delta}(\tilde{x}_i)\,,$$

where $\tilde{x}_i$ is x_i with 2Δ symbols truncated on the left and Δ on the right. This is the case because if the MHW is w' at the beginning of a 1-round, then after the first 2Δ slots, it grows to $w' + B$ due to the fact that the boosted block lies on a chain that is dominant at the beginning of the round, and by virtue of 1-rounds, the first honest party sees the certificate during the initial Δ rounds. Due to the $\tilde{x}_i$-portion of the round, the MHW grows by at least $h_{\Delta}(\tilde{x}_i)$ by the end of the round.

The MHW increase during the ?-period, the subsequent HCI period, and the remaining slots until slot s is always at least $h_{\Delta}(\tilde{y})$, where $\tilde{y}$ is y with Δ slots truncated on each side. This follows from the fact that the potential release of the certificate from the ?-round can only improve the MHW.

Towards (II), first, observe that between slot s_{left} and the end of r_{left}, chain C may gain (on top of w) at most weight $\Delta + U$ in blocks; further, it may gain additional weight B from the certificate in r_{left}. During the $t \geq 0$ 1-rounds $r_{\mathsf{left}} + 1, \ldots, r_{\mathsf{left}} + t$, C may gain additional weight at most

$$tB + \sum_{i=1}^{t} \#_{[\mathsf{a}]}(x_i)\,.$$

Finally, during the ?-period, the subsequent HCI period, and the remaining slots until slot s, C gains at most $B + \#_{[\mathsf{a}]}(y)$ additional weight.

Therefore, in order to be dominant in slot s_{right}, it is necessary that

$$\sum_{i=1}^{t} \#_{[\mathsf{a}]}(x_i) + \#_{[\mathsf{a}]}(y) + (B + C + U + \Delta) \;\geq\; \sum_{i=1}^{t} h_{\Delta}(\tilde{x}_i) + h_{\Delta}(\tilde{y})\,,$$

which contradicts GLS condition (12) (cf. Definition 17).

In case (B), let y_{pre} be the portion of the leader string from slot $\max(s_{\mathsf{left}}, s')$, where s' is the first round of r_{left}, to the last slot just before the HCI period, y_{hci} the leader string during the HCI period, and y_{post} the leader string after the HCI period until slot s. Note that

$$z \;:=\; \underbrace{y_{\mathsf{pre}} \| y_{\mathsf{hci}} \| y_{\mathsf{post}}}_{y}$$

satisfies the five conditions of Definition 17.

Towards (I), the MHW by s_{right} is at least $w + h_{\Delta}(\tilde{y})$, where $\tilde{y}$ is y with Δ slots truncated on each side. Towards (II), observe that chain C may gain weight at most $B + \#_{[\mathsf{a}]}(y) + \Delta$ until slot s_{right} (the Δ is relevant when $s_{\mathsf{right}} < s'$).

Therefore, in order to be dominant in slot s_{right}, it is necessary that

$$\#_{[\mathsf{a}]}(y) + (B + \Delta) \;\geq\; h_{\Delta}(\tilde{y})\,,$$

which contradicts GLS condition (12) (cf. Definition 17).

To conclude the base case, observe that the first part of Lemma 6 implies that all chains dominant at any point after the first cycle c_1 thus contain an honest block in the HCI period of c_1.

The induction step follows along similar lines, but anchoring the argument at the honest block known to exist in the HCI period of the previous cycle. Specifically, assume the lemma holds up to cycle c_{n-1}. Then, to establish the lemma for c_n, inductively identify an honest block B in the HCI period of c_{n-1} and redo the above argument, adding the portion of the leader string between B and the end of the last 0-round to the GLS property as x_{pre} (cf. Definition 17).

Round number of last certificate on chain. The second helper lemma characterizes the round number of the latest certificate on chain at the end of a cooldown-shaped suffix of the voting string.

Lemma 7. *r Consider an execution spanning exactly n rounds and assume a good leader string is chosen. Further, assume that the execution yields a valid voting string $\sigma \in \{0, 1, ?\}^n$ of the form*

$$\sigma \;=\; \ldots\; 1\; ?\; 0^{L'}\; (\; ?\; 0^{K-1}\;)^{v}\; ?\; 0^{L}$$

for $L' \in \{K-2, K-1\}$, $v \geq 0$, and $L \geq 0$. Then, there is an integer $c \geq 0$ such that for all parties P, $\mathsf{round}(\mathsf{cert}^{*}_{P,n+1}) = n - L - cK$.

Proof. Let r_1 be the last 1-round in σ and $r_?$ be the ?-round immediately following r_1; let $r'_?$ be the first ?-round after $r_?$.

First, note that since there is a complete CI period following r_1, Lemma 6, Part 2, (with $M = n$) implies that the certificate from round r_1 appears on the chain of every honest party at the beginning of round $n + 1$; thus, $\mathsf{round}(\mathsf{cert}^{*}_{P,n+1}) \geq r_1$.

Assume now the lemma is false. It follows immediately that any potential certificates from $r'_?$ and all following ?-rounds cannot be $\mathsf{cert}^*_{P,n+1}$ since they would satisfy $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = n - L - cK$ for some integer $c \geq 0$. Therefore, the only options left are $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) \in \{r_1, r_?\}$.

Furthermore, observe that $r'_? - r_? \geq K - 1 = T_{\mathsf{HCI}} + TCS$, which means, by Lemma 6, Part 1, that at the beginning of round $r'_?$ the inclusion window for the certificate from $r_?$ was in the settled part of the ledger. Thus, $\mathsf{cert}^*_{P,r'_?}$ was the same for all P and could not have changed until round $n+1$, i.e., $\mathsf{cert}^*_{P,r'_?} = \mathsf{cert}^*_{P,n+1}$.

The assumption that the lemma is false now implies that

- either $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = \mathsf{round}(\mathsf{cert}^*_{P,r'_?}) = r_1$ and $L' = K - 1$
- or $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = \mathsf{round}(\mathsf{cert}^*_{P,r'_?}) = r_?$ and $L' = K - 2$,

both of which violate VR-2B. □

Proof of the voting string theorem. We are now ready to prove the voting string theorem.

Proof (of Theorem 4). The proof is by induction on the length of σ. The base case for $|\sigma| = 1$ follows from rule (I) since $\sigma_1 = 1$ by definition.

Assume now the theorem holds for $|\sigma| = n$. One must show that σ_{n+1} follows the HS rules. To that end consider the following case distinction:

- **Case $\sigma_n = 1$:** In this case, at least one honest party saw a round-n certificate by the end of round n and will consequently, due to VR-1, cast a vote in round $n+1$. Therefore, $\sigma_{n+1} \in \{?, 1\}$, which follows HS-II.
- **Case $\sigma_n = ?$:** In this case, the only permissible extension of σ is $\sigma_{n+1} = 0$, using HS-III. To show this, first observe that $\sigma_n = ?$ implies that no honest party has seen a round-n certificate by the end of round n, and therefore, VR-1 is false for all of them at the beginning of round $n+1$. Furthermore, by the induction hypothesis, the HS rules preclude $\sigma_{n-1} = ?$. Thus, consider the following two subcases:
  - *Case $\sigma_{n-1} = 1$:* At least one honest party must have seen a round-$(n-1)$ certificate by the end of round $n-1$. Since $U \geq \Delta$, that certificate will be delivered to all honest parties before the beginning of round $n+1$, and therefore, $\mathsf{round}(\mathsf{cert}'_{P,n+1}) \geq n-1$ for all honest P.
    Therefore, using that $R > 2$, VR-2A is false for all honest parties in round $n+1$. This implies that $\sigma_{n-1} = 0$.
  - *Case $\sigma_{n-1} = 0$:* Using the induction hypothesis, the HS rules imply

    $$\sigma \;=\; \ldots\; 1\; ?\; 0^{L'}\; (\; ?\; 0^{K-1}\;)^v\; ?$$

    for some $L' \in \{K-1, K-2\}$ and $v \geq 0$. Using Lemma 7 with $L = 0$, one obtains that $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = n - cK$ for some $c \geq 0$ for all honest parties P. Therefore, $n+1-\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = cK+1$, which means that no honest party votes in round $n+1$. Thus, $\sigma_{n-1} = 0$.
- **Case $\sigma_n = 0$:** Again, observe that $\sigma_n = 0$ implies that no honest party has seen a round-n certificate by the end of round n, and therefore, VR-1 is false for all of them at the beginning of round $n+1$.

By the induction hypothesis and the HS rules, $\sigma = \ldots ?\ 0^L$ for $1 \leq L \leq K-1$. Consider the following subcases depending on L:

- *Case $L < K-2$:* In this case, the only permissible extension of σ is $\sigma_{n+1} = 0$, using HS-IV or HS-VI. To show this, consider the following further subdivision:
  - *Case $\sigma = \ldots 1\ ?\ 0^L$*: Let r_1 be the last 1-round in σ. Another case distinction is necessary to distinguish cases where VR-2A or VR-2B is applicable and to show $\sigma_{n+1} = 0$, which follows HS-IV:
    - *Case $L + 2 < R$:* Observe that the certificate from r_1 was delivered to all honest parties by the beginning of round $n+1$ and thus, $\mathsf{round}(\mathsf{cert}'_{P,n+1}) \geq n + 1 - (L + 2)$. Consequently, $n + 1 - \mathsf{round}(\mathsf{cert}'_{P,n+1}) \leq L + 2 < R$, which falsifies VR-2A for all P. Hence, $\sigma_{n+1} = 0$.
    - *Case $L+2 \geq R$:* Since $n+1-r_1 = L+2 \geq R$ and $R, A \geq T_{\mathsf{HCI}}$, Lemma 6, Part 2, (with $M = n$) implies that the certificate from r_1 is on every chain held by an honest party at the beginning of round $n + 1$, i.e., $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) \geq r_1$ for all P. Since therefore, $n + 1 - \mathsf{round}(\mathsf{cert}^*_{P,n+1}) \leq L + 2 < K$, VR-2B is false and no P votes in round $n + 1$. That is, $\sigma_{n+1} = 0$.
  - *Case $\sigma = \ldots 0\ ?\ 0^L$*: Using the induction hypothesis, the HS rules imply

    $$\sigma \ = \ \ldots\ 1\ ?\ 0^{L'}\ (\ ?\ 0^{K-1}\ )^v\ ?\ 0^L$$

    for some $L' \in \{K-1, K-2\}$ and $v \geq 0$. Using Lemma 7, one obtains that $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = n - L - cK$ for some $c \geq 0$ for all honest parties P. Therefore, $n + 1 - \mathsf{round}(\mathsf{cert}^*_{P,n+1}) = n + 1 - (n - L - cK) = cK + L + 1$, which is not a multiple of K since $2 \leq L + 1 < K - 1$. Hence, VR-2B is false and no honest party votes in round $n + 1$ and $\sigma_{n-1} = 0$, which follows HS-VI.
- *Case $L = K - 2$:* Consider the same two subcases as in the previous case:
  - *Case $\sigma = \ldots 1\ ?\ 0^L$*: According to the HS rules, any symbol is acceptable as σ_{n+1} and hence, there is nothing to show.
  - *Case $\sigma = \ldots 0\ ?\ 0^L$*: Similarly to above, using Lemma 7, one obtains that $n + 1 - \mathsf{round}(\mathsf{cert}^*_{P,n+1}) = cK + L + 1$, which is not a multiple of K since $L + 1 = K - 1$. Hence, VR-2B is false and no honest party votes in round $n + 1$ and $\sigma_{n-1} = 0$, which follows HS-VI.
- *Case $L = K - 1$:* Let r_1 and $r_?$ be the last 1-round and ?-round in σ, respectively. Note that $\mathsf{cert}'_{P,n+1} \leq r_?$ and thus, clearly, $n + 1 \geq \mathsf{cert}'_{P,n+1} + R$, which means that VR-2A is satisfied for all honest parties. One final time, consider the same two subcases as in the previous cases:
  - *Case $\sigma = \ldots 1\ ?\ 0^L$*: Observe that $n - r_? = K - 1 = T_{\mathsf{HCI}}$, $\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = \mathsf{round}(\mathsf{cert}^*_{P,n})$ (using Lemma 6, Part 1), and, moreover, $\mathsf{round}(\mathsf{cert}^*_{P,n}) \geq r_1$ since $A \geq T_{\mathsf{HCI}}$. Since $\sigma_n = 0$, no honest party voted in round n. This is only possible if the certificate from $r_?$ was already on the chain in round n, i.e., $\mathsf{round}(\mathsf{cert}^*_{P,n}) = r_?$. Hence, VR-2B is satisfied for all honest parties. It follows that all honest parties vote in round $n + 1$ and $\sigma_{n+1} \in \{?, 1\}$, which follows HS-V.
  - *Case $\sigma = \ldots 0\ ?\ 0^L$*: Similarly to above, using Lemma 7, one obtains that $n+1-\mathsf{round}(\mathsf{cert}^*_{P,n+1}) = cK+L+1$, which *is* a multiple of K since $L+1 = K$.

Hence, VR-2B is satisfied for all honest parties and all honest parties vote in round $n+1$ and $\sigma_{n+1} \in \{?, 1\}$, which follows HS-VII.

□

A.10 Concluding the Consistency Proof

Theorem 5. *The Peras protocol satisfies consistency except with negligible probability.*

Proof. Assume a good leader string is chosen, which fails to happen only with negligible probability, according to Lemma 5. Let C be the chain output by some honest party P in slot ℓ. It suffices to show that C is a prefix of the preferred chain $C_{\mathsf{pref},\mathsf{P}'}$ of any party P' in any slot $\ell' \geq \ell$.

To that end, let ℓ^* be the pruning slot as described in Figure 1. Consider the following two cases, corresponding to the two methods of pruning, where $\rho \in \{0,1\}^*$ is the string such that $\rho_i = 1$ if and only if a round-i certificate has been seen by P:

1. $\ell^* + D$ is followed by at least $\lceil T_{\mathsf{CS}}/B \rceil$ complete rounds i with $\rho_i = 1$: in this case, starting at most at κ_1, the margin relative to ℓ^* reaches $-\kappa_2$ by the end of the happy periods.
2. ℓ^* is followed by at least $T_{\mathsf{CS}} + K$ complete rounds i with $\rho_i = 0$: in this case ℓ^* is followed by an entire CS period of a cooldown, at which point it reaches $-\kappa_2$.

In either case, an argument similar to that in the proof of the first part of Lemma 6 shows that the margin relative to ℓ^* remains negative forever. This in conjunction with Lemma 1 yields the desired claim. □

A.11 Liveness

Liveness of Peras follows from two complementary arguments depending on the shape of the voting string. During cooldown (sequences of 0-rounds), liveness is a direct consequence of Lemma 6. During happy periods (sequences of 1-rounds), a separate chain-quality argument is needed. The parameter T_{HL} denotes the minimum number of consecutive 1-rounds required to guarantee that every dominant chain contains an honest block—analogously to T_{HCI} for cooldown periods.

Theorem 6. *The Peras protocol satisfies liveness with parameter $u = O((T_{\mathsf{HCI}} + T_{\mathsf{HL}})U)$ except with negligible probability.*

Proof. Assume a good leader string is chosen. Let ℓ be some slot and assume that the environment provides a transaction for inclusion to all honest parties in every slot until slot $\ell + u$. It suffices to prove that every chain held by an honest player after $\ell + u$ contains at least one honest block in the interval $[\ell, \ell + u]$.

First, if ℓ is followed by T_{HCI} complete 0-rounds, the second part of Lemma 6 directly implies that there is an honest block in $[\ell, \ell + u]$, based on the fact that the leader string has $(T_{\mathsf{HCI}}, B, D, U)$-cooldown-chain-quality.

Otherwise, it takes at most $O((T_{\mathsf{HCI}} + T_{\mathsf{HL}})U)$ slots until ℓ is followed by T_{HL} complete 1-rounds. In this case, an analogous argument to that of the second part of Lemma 6 shows that there is an honest block in $[\ell, \ell + u]$, the key modification being the following. During 1-rounds, certificates are seen by all honest parties, so both the MHW and any adversarial chain gain B per round from certificates—these contributions cancel. The argument thus

reduces to comparing block-level growth: honest growth is counted during the "effective" portion of each round, i.e., between 2Δ slots after the round start (by which time the certificate has reached all honest parties) and Δ slots before its end (as parties vote for a block on a dominant chain whose depth reflects the state at that point)—exactly the truncated portions $\tilde{x}_i$ in Definition 18. An adversarial chain, on the other hand, can grow by at most $\#_{[\mathsf{a}]}(x_i)$ blocks per round. Note that, unlike in the cooldown case, there is no ?-round whose certificate the adversary could hold "up its sleeve" for an extra B advantage, which is why B does not appear in the slack term of Definition 18. The (T_{HL}, D, U)-happy-chain-quality property of the leader string then ensures that the cumulative honest growth exceeds the adversarial growth, which means that every dominant chain must contain at least one honest block from the T_{HL} consecutive 1-rounds following ℓ. □

A.12 Dynamic-Stake PoS and Bootstrapping from the Genesis Block

A static-stake "longest-chain" (that is, heaviest chain) PoS protocol as the one analyzed so far can be used to bootstrap a PoS protocol with dynamic stake. Since we established the combined security of Ouroboros with a fast settlement feature, the same construction can be used, as the main analytical quantities that establish the security of Ouroboros, namely reach and margin from Definition 13, have been re-proven for our construction to establish consistency.

The lifting to the dynamic stake proceeds as follows: multiple rounds are combined into *epochs*, each of which contains $\mathsf{epLen} \in \mathbb{N}$ rounds. The epochs are indexed by $j \in \mathbb{N}$. During epoch j, leader election is based on the stake distribution $\mathbb{S}_j$ recorded in the blockchain up to g rounds before the beginning of this epoch (in [1,10] the value $g = R$ is chosen). The *epoch randomness* for epoch j is derived as a hash of the additional VRF-values that were included into blocks up to h rounds before the beginning of this epoch (in [1,10] the value $h = R/3$ is chosen).

Security of the full protocol. Lifting the security argument to the full protocol requires additional reasoning to account for the inductive epoch structure. For ease of exposition, we first consider the case $(g, h) = (R, R/3)$.

As depicted in Figure 2, each epoch then consists of three *phases*: Phase 1 must ensure stabilization of the stake distribution for the next epoch, which is taken from the last block before this first phase starts—this block must be stable before the phase ends. This property is argued via a combination of the standard chain-growth (CG) and common-prefix (CP) blockchain properties. Phase 2 guarantees the inclusion of an honest block within this phase in any surviving chain, and relies on the existential chain quality ($\exists\mathsf{CQ}$) guarantee. Finally, the role of Phase 3 is to stabilize all the randomness-determining blocks (those belonging to the rounds of Phase 2) via the same combination of CG and CP arguments as the first phase. We refer an interested reader to [1, Theorem 7, full version] for details of this argument.

The logic of the argument remains unchanged in the general case: if $B_{<t-g}$ and $B_{<t-h}$ denote, respectively, the last blocks affecting the stake distribution and randomness to be used in some epoch e, then the following two conditions are required by the inductive argument:

(C1) $B_{<t-g}$ must become stable sufficiently early to allow for a guaranteed honest contribution to the randomness for e after that; and

(C2) $B_{<t-h}$ must become stable by the beginning of the epoch e.

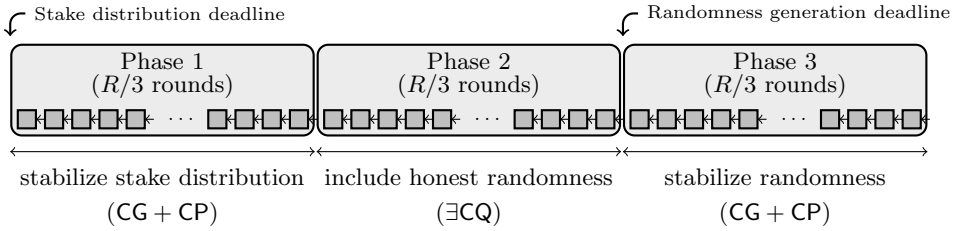


Fig. 2. Illustration of the three phases of an epoch for the inductive argument underlying Ouroboros Genesis in the case $(g, h) = (R, R/3)$.

Weight-based Genesis Rule. Ouroboros Genesis implements a secure procedure for parties to join the execution later only knowing the correct genesis block using the so-called *Genesis rule*: Joining parties determine their state by listening to the broadcast chains for sufficiently long and applying a specific rule to choose the one they adopt as their state. Namely, for any pair of two competing chains, if they branch in very recent past the longer one is preferred; but if they branch in more distant past, a joining party gives preference to a chain that is more dense (i.e., contains more blocks) in a fixed period of gw rounds after the branching point of these two chains (where gw is a protocol parameter chosen in the order of the security parameter(s)). Recall that honest parties sign off blocks as well as votes for certificates using key-evolving signatures to tame adaptive corruptions.

Not surprisingly, our *adjusted Genesis Rule* simply states that density is not measured in blocks but more generally in weight, which is accumulated by blocks (weight of 1) as well as certificates (additional weight B for a certified block).

Definition 20 (Weight-based Genesis Rule). *Let P be a party. For any pair of two competing chains C_1 and C_2, let B be their last common block in slot s and consider the set of slots $I_{\text{gen}} = [s+1, \ldots, s+\mathsf{gw}]$ of size gw. Party P gives preference to the chain that carries more accumulated weight on interval I_{gen} in the local view of P, where weight accumulation happens by means of blocks with slot numbers $s' \in I_{\text{gen}}$, whereby each block has weight 1 and any known certificate certifying a block in interval I_{gen} adds an additional weight B.*

This chain preference rule implies that a party P is following the heaviest chain unless the diverging slot is more than gw slots away from the current time. This also implies that the last common block must be buried under weight in the order of the security parameter on both candidate chains.

Security proof. The core of the security argument is an adaptation of [1, Theorem 2] to our new setting. Intuitively, the theorem asserts that the chain honest parties maintain (ignoring new joiners), is denser in any window of length gw than any chain that could be forged by an adversary.

Lemma 8. *Consider an execution of the dynamic-stake PoS protocol with the weight-based Genesis Rule and let κ denote a security parameter. Consider the first slot $\widehat{\ell}$ in which some honest party P (present since the beginning of the execution) received candidate chain $\widehat{C}$ heavier than its locally adopted chain C_{loc} at the onset of slot $\widehat{\ell}$ and assume the two chain fork where the last common block B has slot $\ell^* < \widehat{\ell} - \mathsf{gw}$. Then, the honest party P discards $\widehat{C}$ except with negligible probability.*

Proof. We first recall that the weight accumulation in the interval of length gw after slot ℓ^* accumulates by two mechanism. Each successful voting round for a boost of a vertex v with $\mathsf{I}_{\#}(v) > \ell^*$ adds at least weight $B >> U$ to the chain, where, for the sake of concreteness, we can assume $B = \kappa/c$ for a constant c. Furthermore, these heavy blocks must be on the same chain corresponding to the sequence of 1-rounds (if any) aligned with slots $\ell > \ell^*$. Also recall that the blocks for each certificate are not necessarily distinct. The second weight increase stems from blocks alone.

For the sake of reaching a contradiction, assume now that P accepts $\widehat{C}$ as its preferred chain due to the condition $\mathsf{Wt}_P(\widehat{C}[\ell^* + 1, \ell^* + \mathsf{gw}]) > \mathsf{Wt}_P(C_{loc}[\ell^* + 1, \ell^* + \mathsf{gw}])$. From the results in Appendix A.9 we see that over the course of κ/U slots, we obtain at least chain growth on the interval of size gw on C_{loc} in the order of $\Omega(\kappa)$, i.e., adopting the candidate chain $\widehat{C}$ indeed causes a rollback of at least weight κ if adopted by P by definition of ℓ^*. By the above condition this implies that the candidate chain $\widehat{C}$ must have accumulated at least as much weight on the same interval.

It remains to argue that we can construct a consistency violation in this execution in the order of κ. Let us first define the following two quantities:

- Let s_1^* be the slot corresponding to the first honestly generated block in C_{loc} with slot number strictly greater than $\ell^* + \mathsf{gw}$; otherwise $s_1^* := \widehat{\ell}$.
- Let s_2^* be the first slot with slot number strictly greater than $\ell^* + \mathsf{gw}$ and associated with a block for which at least one honest party holds a certificate. If no such block exists, let $s_2^* := \widehat{\ell}$.

We let $\widehat{s} := \min\{s_1^*, s_2^*\}$ and make a case distinction:

1. $\widehat{s} = \widehat{\ell}$: This case implies that C_{loc} after the genesis window and up to the current time $\widehat{\ell}$ has no honest block and no certificates are known to P to boost any of these blocks (notice that no certificates could have been formed for a block of slot $\widehat{\ell}$ at this point). This implies that we can build a dominant chain from the competing chain $\widehat{C}$ restricting the characteristic string to slots $0 \dots \widehat{\ell}$: we extend $\widehat{C}$ starting after $\ell^* + \mathsf{gw}$ with a block for any slot s, with $\ell^* + \mathsf{gw} < s < \widehat{s}$, when the slot leader is adversarial. Let's call this extension $\widehat{C}'$. Since the only weight increase until (and including) $\widehat{\ell}$ on C_{loc} is by blocks only, the constructed extension witnesses a consistency violation of order $\Omega(\kappa)$ since $\mathsf{Wt}_P(\widehat{C}[0, \widehat{s} - 1]) > \mathsf{Wt}_P(C_{loc}[0, \widehat{s} - 1])$ implies that $\widehat{C}'$ must be a dominant chain at slot $\widehat{s} - 1$ and the two chains fork at a block buried by at least weight κ. Dominance follows by the fact that either the block for slot $\widehat{\ell}$ to C_{loc} is honest and thus any chain with at least that weight is dominant, or otherwise we can further extend $\widehat{C}$ to be heavier than C_{loc} by appending one more block. It remains to argue that the slot leadership on both candidate chains is identical. This follows by choosing gw as a fraction of an epoch such as $R/6$, where the epoch is a constant multiple of κ. This concludes the first case.
2. $\widehat{s} < \widehat{\ell}$: The condition implies that after the genesis window and up slot number $\widehat{s} - 1$, C_{loc} cannot contain an honest block and no certificates are known to P to boost any of these blocks associated with slots up to $\widehat{s} - 1$. From the existence of either an honest block or a certificate for a block with slot number $\widehat{s}$ we again extend the competing chain $\widehat{C}$ restricting the characteristic string to slots $0 \dots \widehat{\ell}$: we extend $\widehat{C}$ starting after $\ell^* + \mathsf{gw}$ with a block for any slot s, with $\ell^* + \mathsf{gw} < s < \widehat{s}$, when the slot leader is adversarial. We again argue that this extension of $\widehat{C}$ is dominant in the execution up to including slot $\widehat{s} - 1$, which implies a consistency violation in the order of κ as above.

If slot $\widehat{s}$ is associated with an honest block on C_{loc}, then an honest party must have extended a chain at most as heavy as the constructed competing chain, proving its dominance.
If slot $\widehat{s}$ is associated with a certified block B^*, then some honest party must have had block B^* on its heaviest chain at the time of voting. Therefore, define the pair (P', s') as follows: P' is the first honest party to adopt a chain containing B^* and this chain is adopted in slot s'. Clearly $\widehat{s} \leq s'$ and by definition of $\widehat{s}$, when P' decided to adopt the chain, no certificate for B^* has been formed. We can thus conclude the dominance of our constructed competing chain by observing that we can repeat the above argument for the first slot in $[\widehat{s}, \ldots, s']$ that is associated with an honestly generated block on C_{loc}, if any, and otherwise, our competing chain is dominant in slot s' if none of the blocks until including s' are associated with honest blocks on C_{loc}.
This concludes the statement. □

As shown in [1], the security for newly joining parties follows from the above statement in a modular way: suppose we have a newly joining party P^* in slot s^*. We compare the behavior of P^* to a so-called "virtual party" $\widehat{P^*}$ that is observing the network since the beginning and after slot s^*, the virtual party receives exactly the same messages as P^*. We consider the first chain C_{sync} the virtual party receives and adopts after s^*. Let's call this slot s_{sync}.

We know that if party P^* adopts C_{sync} too, then it has safely joined the network, since the virtual party enjoys the security guaranteed by Lemma 8. Hence, assume now for the sake of contradiction that P^* receives C_{sync} but unlike its virtual counterpart, it discards it due to another chain C_1 held at that point in time. First, notice that both parties are aware of both chains by definition. We can now make a case distinction:

- If the virtual is adopting C_{sync} when it is holding C_1 too at slot s_{sync}, this implies that C^* could not have won the Genesis comparison as defined in Definition 20.
- If the virtual party is adopting C_{sync} in slot s_{sync} when holding a different chain than C, then at some time prior to s_{sync} it must have adopted some chain C_2 that has been preferred to C_1, either because C_1 has been received and discarded in favor of C_2, or because C_2 was adopted replacing C_1. Therefore, C_2 wins the Genesis comparison against C_1, and we know that C_{sync} wins the Genesis comparison against C_2. Hence, the only reason why P^* can discard C_{sync} is because C_1 wins the Genesis comparison against C_{sync}. This constellation necessarily leads to a contradiction: consider the forking points of the three chains: let s_{12} be the slot of the last common block of chains C_1 and C_2 and let $s \leq s_{12}$ be the slot of the last common block of all three chains in question. If $s \geq s_{sync} - \mathsf{gw}$, then C_{sync} must be the longest chain among them and the comparison is transitive, hence C_1 could not have been preferred. If $s < s_{sync} - \mathsf{gw}$, P^* could only have accepted C_1 if C_1 wins the Genesis comparison to C_{sync}, which implies that the virtual party would actually also prefer C_1 over C_{sync}. This which contradicts Lemma 8 since $s < s_{sync} - \mathsf{gw}$ and the virtual party is present from the beginning.

A.13 On Self-Healing in the Presence of the Voting Overlay

It is known that Nakamoto-style PoS can be instantiated to be self-healing with budget $\mathcal{B}$ [2]. It is straightforward to show that in the same sense, our protocol can be instantiated to be self-healing with budget $\mathcal{B}$. Recall from Section 2 that we only consider spike-attacks as resulting from low honest participation leaving a relative majority to the adversary. In

such a scenario, (voting) committees are never adversarially dominated (with overwhelming probability) since the absolute majority is still in the hands of honest (but offline) participants.

Theorem 7. *Consider an execution of the protocol with adjusted cooldown parameters* $T'_{\mathsf{HCI}} = \Theta(T_{\mathsf{HCI}} + \mathcal{B})$ *and* $T'_{\mathsf{CS}} = \Theta(T_{\mathsf{CS}} + \mathcal{B})$ *in the order of the the anticipated spike-budget* $\mathcal{B}$ *and let slot* ℓ^* *be the first slot such that the spike-budget is fully spent by* ℓ^**. Then, with overwhelming probability, the protocol with the adjusted parameters is self-healing with budget* $\mathcal{B}$*.*

Proof (Proof Sketch.). Consider the first round r^* where the honest majority condition is restored. We make a case distinction.

- r^* **is during a cooldown.** For the adjusted parameters as in the statement above, we now argue that we can re-establish the requirements needed for a safe restart as in Lemma 6. To this end, we need to guarantee certificate inclusion by T'_{HCI} as well as settlement on the certificate at the end of the cooldown. We can do this via the following case analysis:
  1. Let r be the smallest slot number such that no older certificate can be included any longer in any block with slot number r or higher, and let $r^* < r$.
     If $r - r^* > \mathcal{B} + T_{\mathsf{HCI}}$, we can heal until the restart by [2], since we are running a Nakamoto-consensus for sufficiently long to absorb the additional spike and have the time window T_{HCI} as before to achieve chain quality. Furthermore, settlement by the end of the cooldown period happens within T_{CS} slots. Otherwise, we know we are less than $\mathcal{B} + T_{\mathsf{HCI}}$ slots away from the certificate inclusion. For an appropriate choice of a cooldown duration in $\Theta(\mathcal{B} + T_{\mathsf{HCI}})$, we are guaranteed that the spike-attack starts sufficiently distant after the start of the cooldown and at a point where the certificate has been included by after T_{HCI} since the start of the cooldown. Furthermore, by [2], any established chain-quality property on that portion of the Nakamoto-chain will not be affected by the spike since the vulnerability window does affect the start of the cooldown. Furthermore, we know that at least T'_{CS} slots are do be executed until the end of the cooldown, in which case, again by [2], since we are running a Nakamoto execution as argued in Lemma 6, it can heal from the additional spike and subsequently settle normally within T_{CS} as in Lemma 6.
  2. In the other case, we are at an advanced state of the cooldown period. Similar to above, chain quality up to the certificate inclusion deadline is not jeopardized by the later spike and what is left to argue here is that settlement is achieved at the end of the cooldown. This follows by an analogous case distinction as above depending on whether the spike happens close to the end of the cooldown, at which point its effect cannot revert the chain back to the point of the certificate-inclusion deadline, or whether the spike happens distant to the end of the cooldown, in which case the last segment of the cooldown period is able to heal and settle as we are in a normal Nakamoto-execution to reach margin of $-\kappa_2$ by definition of T_{CS}.
- r^* **is during a sequence of 1-rounds.** We know from Theorem 1 that margin decreases during 1-rounds since B still dominates the total number of active slots (cannot increase during a spike-attack since the spike is formed by reduced honest participation). Hence, we either heal as margin hits 0 during this sequence of 1-rounds, or we enter cooldown at which case the above case applies.

This concludes the proof. □

The dynamic-stake case. As shown in [2], the ability to self-heal is impacted by the inductive epoch-structure when lifting the static case to the dynamic case. In particular, the PoS protocol only self-heals against adversaries that are not able to raise margin above a certain level determined by the protocol parameters R, g, h, and gw of Appendix A.12, in which self-healing follows for the full protocol immediately. Thus, qualitatively, given a level of anticipated adversarial dominance, one can choose a parametrization that withstands against attacks of that anticipated strength (but not against a more powerful attack).