Ouroboros Taktikos: Regularizing Proof-of-Stake via Dynamic Difficulty

Aaron M. Schutza[1], Hans Walter Behrens[1], Tuyet Duong[1,2] and J. A. Aman[1,3]

[1] Topl
a.schutza, h.behrens@topl.me
[2] FYEO, Inc.
t.duong@gofyeo.com
[3] Rice University
james.aman@rice.edu

Abstract. In any Nakamoto-style distributed ledger technology, participants must eventually come to a deterministic ordering of proposed extensions as a necessary precondition to consensus. To prevent Sybil attacks, this process encodes a bias toward selecting proposers who commit a limited resource. In proof-of-work (PoW) schemes, block proposals are secured using a hashing mechanism that preferences miners with greater computational power. In Nakamoto-style proof-of-stake (PoS) paradigms, proposers' eligibilities derive from their staked holdings, relying on a thresholding mechanism to fairly select the next block proposer with a bias towards those who have committed more stake. This eligibility threshold controls which parties may propose a block in a given slot, with easier thresholds inducing greater block density. However, higher density also increases forking – periods of uncertainty where consensus remains (temporarily) unsettled. Therefore, the selection of the PoS eligibility threshold critically affects both security and throughput in the associated chain. Previous work relies on static threshold values, which simplifies security analysis. In this work, we extend the static eligibility threshold to a dynamic one, and introduce the concept of a *local dynamic difficulty* mechanism in which thresholds follow a non-monotonic difficulty curve. We implement this mechanism in a novel PoS protocol, Ouroboros Taktikos, where we find that the dynamic regime regularizes slot intervals and improves block throughput. We further propose a security analysis framework for the dynamic regime, using a recurrent Markov chain to establish common prefix, chain growth, and chain quality properties. Our analysis establishes exponential concentration bounds in the bounded-delay, static-stake setting. We employ a novel state-space representation of the staking procedure that may be of independent interest in the security analysis of other protocols. We additionally address the security of our approach in the context of a grinding-type adversary. We compare Ouroboros Taktikos to Ouroboros Praos and show that the addition of local dynamic difficulty improves throughput by $\sim 3\times$ and reduces 99^{th} percentile block latency by $\sim 5\times$ compared to the state of the art.

1 Introduction

In traditional Byzantine state machine replication (SMR), agreement typically relies on a set of fixed, known participants. In contrast, distributed ledger technologies (DLTs) leverage randomness to reach consensus with an unknown list of participants. This randomness establishes proposer eligibility for the consensus round, analogous to leader election in SMR, but may contain subtle bias. Attackers can leverage this bias to undermine agreement in their favor, so 'proof' schemes aim to eliminate or mitigate the issue explicitly. The most well-known of these schemes, the proof-of-work (PoW) approach popularized by Bitcoin [28], uses computational resources to secure proposer selection. Given the high power consumption of the 'work', some authors have expressed concerns [24,30,33] over the sustainability of PoW. Alternatives such as proof-of-stake (PoS) [3,6,9,12,13,16,19,22] address this criticism by drawing randomness from alternative sources and improving computational efficiency. To do so, PoS methods execute local, pseudo-random deterministic trials to establish eligibility, with constraints to limit adversarial influence. Several nodes might share eligibility to extend the chain (a *fork*), or an adversary could propose valid extensions of older chains (the so-called *nothing-at-stake* attack [25]). To address these cases, chain selection rules such as [13] limit the impact of forks, while other supplemental techniques improve bootstrapping [3] or clock synchronization [4].

PoS eligibility commonly adopts the concept of a *slot*, or a global eligibility period predicated on some underlying synchronization mechanism. Usually, time governs these slots via the generalized concept of a *time beacon* [3,5,13,22]. Participants agree on time using existing approaches such as NTP, or via more decentralized schemes as in [4]. This synchronization mechanism allows participants to locally determine a numeric eligibility for a given slot from a verifiable random test associated with that slot. The *eligibility threshold* controls whether and which participants may propose a new block. The percent of slots with valid proposals, the *active slots coefficient* f, plays a role similar to mining difficulty in PoW. By tuning f, PoS approaches can approximate existing PoW eligibility distributions [18] to more closely match their consistency bound.

In the Ouroboros family of protocols [22], honest participants discard blocks from future slots, and only share their most recent blocks; thus, the time synchronization constraint protects honest chain growth. Slot frequency typically dominates block frequency, so not every slot produces an eligibility. When the block time interval falls too low, forks become common and allow the adversary to undermine consensus. In contrast, a too-low block frequency sabotages throughput when no eligible leaders emerge for extended periods. Even a carefully-tuned static f remains susceptible to randomness, with sporadic periods of "feast or famine" arising from the uniform randomness of the eligibility threshold. Furthermore, adversaries can look ahead and test future slots associated with a given epoch nonce to determine the relative value of their private forks. This behavior, known as *grinding*, gives the adversary a small but persistent advantage within the consensus process that scales with the number of forks. We propose to address these challenges by moving from a static coefficient to a dynamic, adaptive one.

1.1 Our Contributions

Proof-of-Stake protocol with local dynamic difficulty. The protocol presented here, *Ouroboros Taktikos*[4], solves these problems and improves upon Ouroboros Praos [13] in several surprising ways. We replace the static active slot coefficient f with a *local dynamic difficulty* (LDD) function $f(\delta)$ we call a *difficulty curve*, where eligibility depends not only on stake but also on a slot interval δ. The stochastic variability introduced by Taktikos obscures forward prediction of leadership eligibility, preserving security characteristics and making covert coordination more difficult even in the context of a grinding adversary.

Implementation and empirical comparison. We further implement our protocol [**][5] and empirically evaluate its performance against existing approaches. Specifically, we compare with Praos [13], noting that the LDD method can adapt to any eligibility game that relies on an active slot coefficient. We additionally evaluate the effects of LDD on our consistency bound under varying adversarial assumptions. We show that our procedure improves upon existing approaches in several metrics, reducing both variability and expected value of the block time interval. In aggregate, these improvements result in a $\sim 3\times$ increase in block throughput, and a $\sim 5\times$ decrease in 99^{th} percentile block latency.

Security framework to analyze PoS blockchain consistency. Finally, we supplement our empirical assessment with a detailed security analysis. We establish a generalized discrete-time Markov model to analyze the dynamics and steady-state behavior of eligibility tests, given an optimal adversary in a (semi-)synchronous environment. This allows us to predict chain growth and security properties for an environment $\mathcal{Z}$ with participants $\mathcal{H}$ (honest) and $\mathcal{A}$ (adversarial), following a difficulty curve $f(\delta)$. We use this framework to compare behavior under local dynamic difficulty with the idealized $\mathcal{G}_{\text{LEDGER}}$ in [3], showing that LDD-enhanced Nakamoto PoS competes with the consistency bound of PoW schemes without the commensurate computational overhead.

2 Design

Local dynamic difficulty. We first briefly recap the process of minting eligibility in Nakamoto-style proof of stake to establish context. Verifiable random function (VRF) outputs consist of a pseudo-random byte-string y of length ℓ_{VRF} and a proof π. As in Ouroboros Praos [13], a potential minter generates a test nonce $y_p \in {}^{\{0,1\}^{\ell_{\text{VRF}}}}\!/_{2^{\ell_{\text{VRF}}}}$, which represents an output from a VRF indistinguishable from uniform randomness in the range $(0, 1)$. An associated public key corresponding to the forging party p can verify each pair (y_p, π_p). We leave this functionality unaltered and model our own nonce generation on [3].

In existing approaches, the test nonce y_p is evaluated by minting parties and validators to elect slot leaders in the forging procedure by checking y_p against a static eligibility threshold. In contrast, we introduce a new eligibility paradigm based on a difficulty curve $f(\delta)$, which allows for a dynamic eligibility threshold. This concept, which we term *local dynamic difficulty* (or LDD), induces

[4] From τᾰκτῐκός, ordering or arranging, especially in a tactical sense.

[5] Some artifacts have been anonymized with ** to preserve blinding.

a dominant distribution with properties influenced by the choice of curve $f(\delta)$. The mechanism of LDD changes the probability of forging blocks based on how recently the previous block was observed. As with a static f, we consider $f(\delta)$ agreed upon by all parties in a global setup. We define the *slot interval* δ as a variable that measures the number of slots between consecutive blocks. More formally, the slot interval δ_ℓ for block B_ℓ is defined as the difference between slot $\mathtt{sl}_\ell$ of B_ℓ and the slot $\mathtt{sl}_{\ell-1}$ of the parent block $B_{\ell-1}$, i.e. $\delta_\ell = \mathtt{sl}_\ell - \mathtt{sl}_{\ell-1}$ where δ is used as a variable to index all possible slot intervals over the domain $\delta > 0$, since $\mathtt{sl}_\ell > \mathtt{sl}_{\ell-1}\, \forall \ell$. We treat $\mathtt{sl}_{\ell=0}$ as the genesis slot.

In the LDD forging procedure, we use a threshold function that satisfies the eligibility test with δ as an argument:

$$\phi(\delta, \alpha) = 1 - \big(1 - f(\delta)\big)^{\alpha} \tag{1}$$

We see that independent aggregation remains valid across each slot interval:

$$1 - \phi\bigg(\delta_\ell, \sum_p \alpha_p\bigg) = \prod_p \big(1 - \phi(\delta_\ell, \alpha_p)\big) \tag{2}$$

A test procedure analogous to the Praos forging procedure is carried out in each slot. A block eligibility is valid if, given nonce η, block B_ℓ in slot $\mathtt{sl}_\ell$, parent block $B_{\ell-1}$ in slot $\mathtt{sl}_{\ell-1}$, and forging party p, the following inequality is true: $y_p(\mathtt{sl}_\ell) < \phi(\delta_\ell, \alpha_p)$.

Before introducing our proposed difficulty curve, we first define several related terms. A *slot gap*, $\psi \geq 0 : f(\delta < \psi) = 0$, requires that no eligibilities may occur closer than ψ slots. This allows the protocol to explicitly consider network delay Δ, bounding $(\Delta < \psi)$ adversarial power derived by front-running block propagation. To compensate for the loss of chain growth induced by a slot gap, we dynamically increase forging capacity. We therefore specify a *forging window* of $\gamma - \psi$ slots where $f(\delta \leq \gamma)$ increases with δ to a maximum *amplitude* of f_A; that is, blocks become easier to mint as more empty slots occur. To retain the security properties of the underlying protocol, during bootstrapping $f(\delta)$ falls to a *baseline difficulty* f_B, independent of δ. We refer to the domain of $\delta > \gamma$ as the *recovery phase* of the LDD forging procedure. The interplay between these three domains may be finely tuned to establish new block dynamics, with completely different block time distributions and security properties.

We present a difficulty curve featuring a slot gap, forging window, and recovery phase defined by the 4-tuple (ψ, γ, f_A, f_B), which we call a *snowplow curve*:

$$f(\delta) = \begin{cases} 0 & \delta < \psi \\ f_A \cdot \left(\frac{\delta - \psi}{\gamma - \psi}\right) & \psi \leq \delta < \gamma \\ f_B & \gamma \leq \delta \end{cases} \tag{3}$$

where $0 \to \psi$ is the slot gap, $\psi \to \gamma$ is the forging window, f_A is the amplitude, and f_B is the baseline difficulty. We constrain these parameters by $0 < f_B \leq f_A \leq 1$ and $0 \leq \psi < \gamma$. The Taktikos curve permits leadership eligibility to be biased by selectively extending tines with parent slots in the forging window $\delta < \gamma$, but

for any $\delta \geq \gamma$ leadership returns to independent predictability [13]. A simplified form of $f(\delta)$ setting $\psi = 0$ and $\delta < \gamma < 1 + \frac{1}{c}$ is defined as:

$$f(\delta) = c\delta \text{ where } c = \frac{f_A}{\gamma} \tag{4}$$

This difficulty curve may be parameterized such that the proportion of blocks having slot intervals in the forging window is arbitrarily close to 1. The distribution of leader election events induced by this curve converges geometrically on $0 < \delta < \frac{1}{c} + 1$ (see Appendix B), and by choosing a $\gamma < \frac{1}{c}$ an arbitrarily small portion of blocks fall in the recovery phase on the honest-majority tine.

Intuitively, this curve rewards well-synchronized nodes with increased block production without requiring additional messaging, while still allowing out-of-date participants to catch up if needed. We now use the snowplow curve to define a revised consensus protocol that leverages LDD for its leader election.

The consensus protocol. Using LDD, we extend Ouroboros Praos [13] to a new protocol, *Ouroboros Taktikos*. Taktikos operates in the hybrid model with the functionalities $\mathcal{F}_{\mathsf{INIT}}$, $\mathcal{F}_{\mathsf{VRF}}$, $\mathcal{F}_{\mathsf{KES}}$, $\mathcal{F}_{\mathsf{DSIG}}$.[6] We emphasize that the modifications we make bring significant performance improvements with similar security guarantees when the difficulty curve is appropriately parameterized. The protocol consists of three phases: the first phase is the *initialization phase* where stakeholders obtain the public keys v_i^{vrf}, v_i^{kes}, v_i^{dsig} from the ideal functionalities $\mathcal{F}_{\mathsf{VRF}}$, $\mathcal{F}_{\mathsf{KES}}$, $\mathcal{F}_{\mathsf{DSIG}}$. In the second phase, the stakeholders register to $\mathcal{F}_{\mathsf{INIT}}$ with their public keys and stake, receive the genesis block, and set it as their local chain. The third phase is the *chain extension* phase where stakeholders mint blocks. We illustrate the relevant steps of Π^{Tak} in Figure 6 in the Appendix, which also highlights key differences relative to Praos.

3 Evaluation

Experimental setup. The shift to LDD represents a substantial change to the eligibility test, the effects of which require careful evaluation. The conditional probabilities inherent in LDD confound exhaustive evaluation because the adversary has a new degree of freedom. Local dynamic difficulty complicates combinatorial treatments such as [8,21]. In lieu of established frameworks, we develop and present a novel theoretic model in Section 4, and create a Monte Carlo-based simulator to directly estimate the distribution of eligibilities. Our simulator [∗∗] encapsulates a superset of eligibility test behavior by adding LDD to Ouroboros Praos, emulating the behaviors of both approaches to provide an appropriate baseline for comparison.

For our chosen difficulty curve, we use $f(\delta) = (0, 15, 0.5, 0.05)$. We chose these parameters to provide similar common prefix and chain quality to [1], while also enhancing chain growth. Note that $\psi = 0$ indicates a network delay of 0 ($\Delta = 0$), an unrealistic choice in practice but suitable for our simulated context. Because $\psi \geq \Delta$ allows for tuning to observed network conditions, we can maximize both

[6] See Appendix A for a version lifted to the semi-synchronous setting, with a modified chain selection rule `maxvalid-tk`.

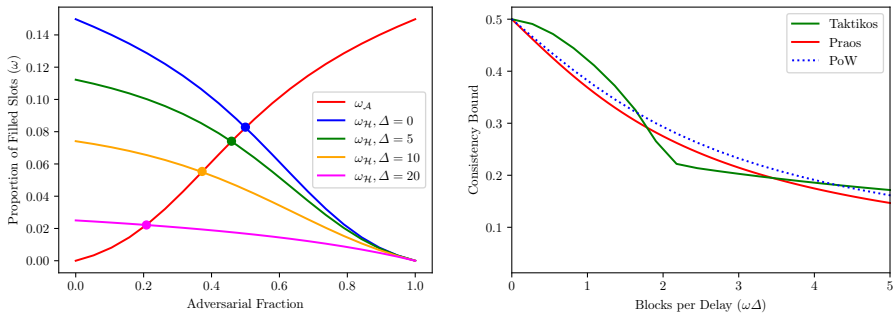

Fig. 1: (a) Effective honest throughput with varying network delay. (b) Consistency bound comparison in terms of blocks per delay interval, with PoW trend from [18].

security and throughput even in the synchronous case. Our initial survey of the LDD parameter space serves as a starting point for future work exploring alternate $f(\delta)$ formulations, which may use curves of arbitrary complexity. All experiments were conducted for 10^5 slots over 30 trials.

Figure 1(a) highlights the critical effect of network delay on *effective honest throughput*, referring to the number of blocks proposed per slot by honest participants, excluding slots in which an adversarial eligibility could supersede it. As Δ increases, out-of-band communication exclusively available to the adversary gives additional power in tie-breaking and frontrunning honest eligibilities. The points in Fig. 1(a) at which $\omega_{\mathcal{H}}$ intersects $\omega_{\mathcal{A}}$ mark the boundary beyond which the adversary produces more blocks than the honest nodes, violating consistency. This shows how long delays can substantially reduce consistency in static Nakamoto-style approaches. Under LDD we may tune $f(\delta)$ to consider delays, which mitigates the effects of network latency. When appropriately tuned, this effect actually results in a positive influence on the consistency bound as shown in Figure 1(b). Intuitively, by rewarding participants in sync with the network and penalizing those operating with a delay (intentional or not), it becomes more difficult for an adversary to intentionally mislead the honest participants for well-tuned configuration spaces.

Examining Figure 1(b) in more detail, we see the consistency bound as a function of block production rate for several protocols. Recall that the consistency bound defines a safe upper limit for the adversarial fraction, beyond which consistency is violated. The tight bound on PoW established by [18] serves as our benchmark for comparison to the literature. Our analytic model for Praos aligns with the behavior from [13], confirming the applicability of our model for this domain. Interestingly, we find that over some regions Taktikos exceeds the consistency bound of the listed protocols for a given block production rate. For example, with a consistency bound of 0.45, the secure block production rate in Praos is approximately equivalent to PoW. At the same level of safety, Taktikos exhibits a meaningfully higher block production rate, suggestive of the potential performance benefits arising from LDD.

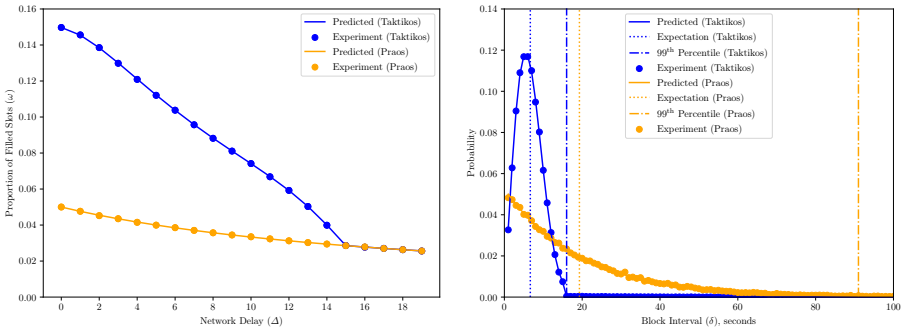

Fig. 2: Experimental comparison between Taktikos and Praos staking procedure. (a) Maximum expected chain growth as network latency increases; higher is better. (b) Probability density distribution of block intervals; lower is better.

Performance. The Taktikos implementation of LDD offers two important improvements over existing approaches. First, it increases throughput by optimizing the usage of slot eligibilities, reducing the amount of time the chain spends idle. Second, it regularizes block production to improve block interval predictability, reducing observed variance. The dramatic effects of these changes are illustrated in Fig. 2, for the chosen parameterization of Taktikos[7] and Praos with a static threshold of $f = 0.05$ based on the real-world value used by [1].

Taktikos provides equal or substantially better effective honest throughput for all values of network delay (see Figure 2(a)). The underlying reason for this advantage arises from the shift of the block interval distribution induced by the linear increase in the snowplow curve (from ψ to γ). In the region $\delta \geq \gamma$, Taktikos enters the recovery phase and performs identically to Praos as demonstrated in Figure 2(a) when $\Delta \geq 15$. Over a range of realistic network latencies, $\Delta \sim 0 - 2$ seconds [29], Taktikos performs $\sim 300\%$ better than Praos. We contend that this increased chain growth is evidence of improved throughput, without the reduction in chain quality typically associated with high block production rates.

Transaction settlement relies in part on the number of blocks observed since that transactions' inclusion, an argument that relates settlement directly with chain growth. That is, as long as honest growth outpaces adversarial growth (the secure regime), the chain asymptotically approaches settlement as it grows [14]. By reducing variance in the block time interval, Taktikos increases predictability of block production and therefore in transaction settlement – a desirable property for many real-world use cases. In Figure 2(b), we see that Taktikos shows a tight distribution of block intervals with mean, variance, and 99^{th} percentile respectively of ($\mu = 6.67$, $\sigma^2 = 17$, 99^{th} percentile $= 16$), where the 99^{th} percentile is the time below which 99% of block intervals fall. All values are given in units of δ. In contrast, Praos shows a much wider distribution, with ($\mu = 19.3$, $\sigma^2 = 318$, 99^{th} percentile $= 91$). Transaction settlement (which depends on block depth) therefore occurs $\sim 5\times$ more quickly and predictably under Taktikos.

[7] $f(\delta) \to (\psi = 0, \gamma = 15, f_A = 0.5, f_B = 0.05)$

Grinding adversaries. At its core, local dynamic difficulty introduces conditional probabilities that enable a new class of grinding attacks on slot eligibilities. Praos does not suffer from the same attack since the protocol determines eligibility independently of the parent block. We refer to the resulting difference in adversarial chain growth in Taktikos as the *grinding frequency*. At each eligibility, the adversary chooses to publicly use or privately save an extension, which influences their future eligibility distribution. However, the adversary may postpone that decision by maintaining both chains locally (a private fork). Consequently, the number of maintained forks scales exponentially at each adversarial eligibility. To predict the grinding frequency for a given difficulty curve, we must test every root-to-leaf path among the maintained tines. This branching structure makes it difficult to formulate an analytical solution, and computationally intractable to check exhaustively. Instead, we adopt a simulation-based approach with a cap on the number of maintained forks at a parametric depth d, which we refer to as the filtration rule. Note that $d = 1$ represents equivalence to honest behavior, where the participant extends only the longest chain.

For the simulated staking procedure, we need only consider the VRF nonce values and the staking threshold function $\phi(\delta, r) = 1 - (1 - f(\delta))^r$, similar to Equation 1. We can further simplify the process by assuming the adversary pools all stake into a single account, a safe assumption due to the independent aggregation property. Conceptually, we draw a series of random variables and then test the thresholds among all possible branches to see which produce valid extensions at each slot. We represent this process in Algorithm 2 for the unbounded, exhaustive evaluation, and Algorithm 3 with the filtration rule.

More formally, let $\mathcal{Y}$ be a set of i.i.d. random variables $y_i \in \mathcal{Y}$ such that $y_i \in [0, 1]$. The set $\mathcal{Y}$ will be an input and the duration of the simulation is $L = |\mathcal{Y}|$ slots. Additionally, we specify as inputs the amount of stake r that the branching adversary controls along with the difficulty curve $f : \mathbb{N} \to [0, 1]$. The grinding frequency is given by taking the limit as $L \to \infty$. The grinding frequency is given by the limit of Algorithm 2 as $L \to \infty$ such that

$$\omega_g = \lim_{L \to \infty} \mathsf{GrindingSim1}(\mathcal{Y}, r, f) \tag{5}$$

This rate of chain growth is given by the leading branch's block number divided by the total number of slots executed by the simulation. The grinding frequency varies as a function of stake that the adversary controls, and is an emergent value given the difficulty curve.

For a practical computation of grinding frequency, we use Algorithm 3 with $d > 1$ and observe that

$$\mathsf{GrindingSim1}(\mathcal{Y}, r, f) = \lim_{d \to \infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f) \tag{6}$$

and as a first-order approximation

$$\omega_g \approx \lim_{L \to \infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f)\big|_{d>1} \tag{7}$$

For comparison with honest chain growth, we predict block frequency by:

$$\omega = C\left[\sum_{\delta=0}^{\infty} \delta \pi_\delta \left(1 - \left(1 - f(\delta)\right)^r\right)\right]^{-1} \tag{8}$$

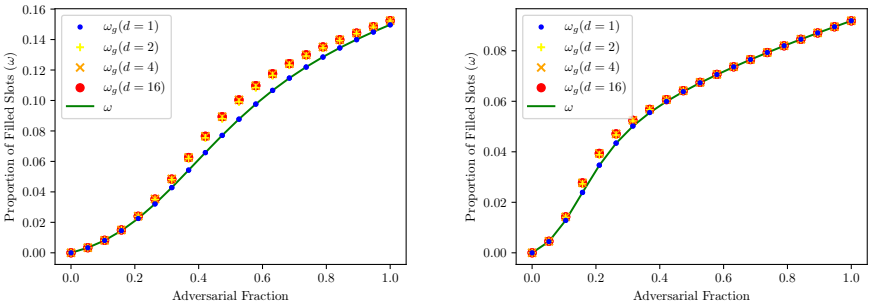

Fig. 3: Grinding advantage by adversarial fraction for (a) $\gamma = 15$. (b) $\gamma = 40$.

where π_δ is the stationary distribution (Eq. 14) and C is a normalization constant:

$$C = \sum_{\delta=0}^{\infty} \pi_\delta \left(1 - \left(1 - f(\delta)\right)^r\right)$$

Figure 3 presents a comparison of grinding frequency with respect to adversarial fraction for varying filtration depths, d. We verify this model by noting the equivalence of block frequency to the predicted grinding frequency when $d = 1$:

$$\omega \approx \lim_{L\to\infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f)\big|_{d=1} \tag{9}$$

The honest production rate ω, is predicted with Equation 8. While the adversarial grinding curves ω_g, are given by evaluating Equation 7 with $L = 10^6$.

We see that as expected when $d = 1$ the grinding frequency ω_g exactly matches the block frequency ω irrespective of γ. We also see that both the magnitude and location of the adversary's benefit shift with γ. As γ increases, the number of potential forks tracked by an adversary also increases, diluting the grinding advantage. As d increases the adversary's memory costs rapidly rise, as the number of tracked branch possibilities increases exponentially. However, the grinding advantage gained from tracking more branches rapidly approaches an asymptote. The benefit between $d = 4$ and $d = 16$ proves negligible despite a much greater cost. Therefore, while Taktikos does provide a small increase in adversarial power (see Appendix C.1), the size of that advantage remains tightly bounded and the other performance gains more than compensate for the tradeoff.

4 Analysis

A new framework for analyzing PoS blockchain consistency. The KRS framework [23] was proposed as an approach for analyzing PoW blockchains, and retains only limited applicability for PoS protocols because the adversary can create many more valid blocks than honest stakeholders due to nothing-at-stake or grinding attacks utilizing costless simulation. Consequently, block-counting approaches do not apply to the PoS setting. Kiayias et al. proposed *forkable string analysis* [22] and counted the number of slot leaders among honest and

adversarial participants to capture these attacks. This approach finds a lower bound on the probability p_0 of a uniquely honest slot leader being elected in each slot, and the upper bound on the probability p_1 of malicious slot leader election. The marginal probability of forkable events in each slot gives the relative influence of nothing-at-stake adversaries purposefully attempting to diverge the network. The forkable string theorem under those constraints states that the normalized $\frac{p_0}{p_0+p_1}$ is bounded below by $\frac{1}{2}(1+\epsilon)$ for $\epsilon \in (0,1)$. Computing the probabilities of forkable events is straightforward when leader elections are independent of the previous blocks, a design choice common to existing Ouroboros protocol specifications. In contrast, the proposed LDD mechanism prevents the assumption of independent trials. To address this, we adopt the KRS Markov framework to compute the probabilities and expectation values of uniquely honest leaders and then integrate forkable string analysis to show blockchain consistency. We emphasize that this new framework enables analyzing a variety of blockchain protocols under several potential attacks and security settings (see Appendix A).

Outline of our framework. We propose a general framework to analyze the consistency of PoS protocols, inspired by the work of KRS [23], defined as follows:

1. Define a Markov Chain with states and *events of interest*, e.g. uniquely honest slot leaders, honest ties, malicious slot leaders, or no leaders.
2. Compute the stationary distribution of the Markov chain and show that it is ergodic and time-homogeneous.
3. Derive expectation values for the events of interest.
4. Apply a concentration theorem using the Chernoff-Hoeffding bounds for generalized Markov chains [11].
5. Apply the *forkable string theorem* [22] to show consistency under the constraints computed from the expectation values and concentration bounds.

4.1 Modeling LDD with a Markov Chain

We model the LDD staking procedure as a discrete-time Markov process M following the formalism presented in [11,23,27]. To correspond to the security setting of [13], we assume a synchronous network and static registration, which fits the model of a time-homogeneous Markov chain. (We lift to the semi-synchronous case in Appendix A .) We design the state space of the Markov process to capture the dynamical evolution of the slot interval as the global slot increments. This requires the assumption that all nodes agree on the slot in which the last block was observed, but may disagree on the block identifier contained in that slot.

Definition 1. ***Characteristic labels.*** The events of M correspond to slot leader configurations in a set of elected parties $\mathcal{P}$ where we label honest participants $\mathcal{H}$ and a static adversary $\mathcal{A}$ executing in environment $\mathcal{Z}$. The possible outcomes of this staking procedure are elections resulting in a unique honest leader $\mathcal{H}_0$, an honest tie $\mathcal{H}_1$, any adversarial leader $\mathcal{A}_1$, or no leader $\perp$. By construction, the events indexed by Λ are correlated with forkable and non-forkable events. The binomial distribution forms the basis of forkable strings. We represent non-forkable events as 0, forkable events as 1, and empty trials with $\perp$ forming another label space $\Lambda' = \{0, 1, \perp\}$.

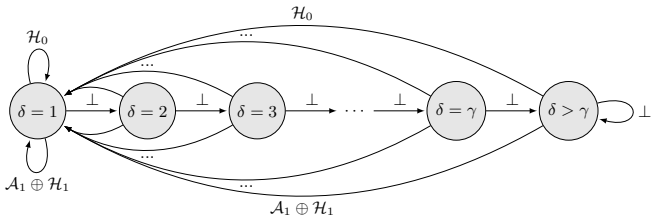

Fig. 4: The Markov process diagram for $\gamma > 0$, assuming that δ increments with each $\perp$ test, where $s_{\delta=1}$ is the replenishing state that the system returns to for every $\not\perp = \mathcal{H}_0 \oplus \mathcal{H}_1 \oplus \mathcal{A}_1$ path taken. For $\gamma > 0$, if the staking procedure depends on δ, then the path probability depends on state.

Definition 2. ***State space*** S ***of*** M**.** We describe discrete counting states to denote the configuration of the leader election process. These states form the basis of random variables that represent the configuration of our Markov chain. Let δ be a discrete variable that is independent of the stake distribution $\mathbb{S}_0$ sampled in M. Consider states s in state space S which are labeled by the discrete values of δ up to a threshold γ such that $s_{\delta=i} \in S_\gamma = \{s_j : 0 < j \leq \gamma\}$. M is a discrete-time event and we ascribe the counting states to the progressive evolution of the variable δ as the staking procedure executes. Once the system has incremented to a configuration where $\delta > \gamma$, the trials no longer depend on δ as M continues to evolve. Let that state $s_{\delta>\gamma}$ correspond to the recovery phase of M. If the system reaches the recovery phase, it remains until any non-empty trial $\not\perp = \mathcal{H}_0 \oplus \mathcal{H}_1 \oplus \mathcal{A}_1$ occurs and returns to the first counting state. With the state space $S = S_\gamma \cup s_{\delta>\gamma}$, M forms a finite-state, irreducible discrete-time Markov process with graph G shown in Figure 4.

We consider the distribution of slot intervals in terms of the process M shown in Figure 4. The probability of no leader ($\perp$) and at least one leader ($\not\perp$) define the transition rates of M under total active stake r as the replenishing process:

$$p_\perp(\delta) = (1 - f(\delta))^r \qquad p_{\not\perp}(\delta) = 1 - (1 - f(\delta))^r \tag{10}$$

Definition 3. ***Transition matrix of*** M**.** The transition matrix for the Markov process M is $\mathrm{P}_M = \{P(i,j) : i,j \in S\}$, where $P(i,j)$ are the transition rates, i.e. the probability for the system to transition from i to j. These rates satisfy

$$P(i,j) \geq 0, \ \sum_{k \in S} P(i,k) = 1 : \ \ i,j \in S \tag{11}$$

Let the states be represented by their numerical indices such that $s_i \leftrightarrow i \in S_\gamma$ and $s_{\delta>\gamma} \leftrightarrow \gamma + 1$. We define the transition rates in terms of the probabilities of leadership election under LDD, and let $b(\delta) = 1 - f(\delta)$. With $p_\perp(\delta) + p_{\not\perp}(\delta) = 1$, we can restructure P_M as a $(\gamma+1) \times (\gamma+1)$ super-diagonal matrix:

$$\mathrm{P}_M = \begin{bmatrix} 1 - b(1)^r & b(1)^r & 0 & ... & 0 \\ 1 - b(2)^r & 0 & b(2)^r & ... & 0 \\ ... & ... & ... & ... & ... \\ 1 - b(\gamma)^r & 0 & 0 & ... & b(\gamma)^r \\ 1 - b(\gamma+1)^r & 0 & 0 & ... & b(\gamma+1)^r \end{bmatrix} \tag{12}$$

4.2 The Stationary Distribution

Definition 4. ***The stationary distribution $\boldsymbol{\pi}$.*** Since P_M is super-diagonal, the form of M corresponds to a backward-recurrent, time-homogeneous Markov chain truncated by an aperiodic term where the time step t corresponds to the number of rounds executed by M [27]. All states in S communicate and the transition matrix is finite and fixed in time. It follows that all states in S are positive recurrent. The stationary distribution $\boldsymbol{\pi}$ is the unique measure obtained as the limiting distribution as $t \to \infty$:

$$\boldsymbol{\pi} = \lim_{t\to\infty} \boldsymbol{u} \cdot \mathrm{P}_M^{(t)} = \boldsymbol{\pi} \cdot \mathrm{P}_M \tag{13}$$

where the input distribution $\boldsymbol{u}$ is a positive definite measure satisfying $\sum_{i\in S} u_i = 1$. The stationary distribution $\{\pi_i : i \in S, \sum_i \pi_i = 1\}$ satisfies the following system of equations given by the transition matrix:

$$\begin{aligned}
\pi_1 &= P(1,1)\pi_1 + P(2,1)\pi_2 + ... + P(\gamma,1)\pi_\gamma + P(\gamma+1,1)\pi_{\delta>\gamma} \\
\pi_2 &= P(1,2)\pi_1,\ ...\ ,\ \pi_\gamma = P(\gamma-1,\gamma)\pi_{\gamma-1} \\
\pi_{\delta>\gamma} &= P(\gamma,\gamma+1)\pi_\gamma + P(\gamma+1,\gamma+1)\pi_{\delta>\gamma}
\end{aligned}$$

This system of equations is solved by row reduction, shown in Eq. 36. Finally, we express the stationary distribution analytically in terms of $f(\delta)$:

$$\pi_1 = \tfrac{1}{C_1},\ \pi_i = \tfrac{1}{C_1}\prod_{j=1}^{i-1}\big(1-f(j)\big)^r,\ \pi_{\delta>\gamma} = \tfrac{1}{C_{\gamma+1}}\prod_{j=1}^{\gamma}\big(1-f(j)\big)^r \tag{14}$$

where $i : 1 < i \leq \gamma$ and

$$C_1 = 1 + \sum_{k=1}^{\gamma-1}\prod_{l=1}^{k}\big(1-f(l)\big)^r + \big[1-\big(1-f(\gamma+1)\big)^r\big]^{-1}\prod_{j=1}^{\gamma}\big(1-f(j)\big)^r$$

$$C_{\gamma+1} = \left(1 + \sum_{k=1}^{\gamma}\prod_{l=1}^{k}\big(1-f(l)\big)^r\right)\left(1-\big(1-f(\gamma+1)\big)^r\right) + \prod_{j=1}^{\gamma+1}\big(1-f(j)\big)^r$$

Estimates of Constant Mixing Time. We use the following theorem to establish that M converges to the stationary distribution with concentration bounds that have guaranteed exponential convergence in time. Depending on how $f(\delta)$ is chosen, different regimes of stability may be induced. The Taktikos difficulty curve described in Section 2 (the snowplow curve) is *strongly ergodic*, in that it acts as two coupled Markov chains with different mixing times. This can be seen as the forging window and recovery phase interacting as two coupled Markov chains with highly distinct convergence rates. In the following, we consider the general case for any $f(\delta)$, so we presently make no assumptions about the type of stability induced by the LDD staking procedure.

Theorem 1. ***Convergence theorem.*** We estimate the convergence rate using the Shannon-McMillan-Breiman Theorem (see [2] for review and proof) and the entropy rate of M at time step t. Given an arbitrary input distribution $\boldsymbol{u}$ and

stationary distribution $\boldsymbol{\pi}$ there exist constants $b \in (0,1)$ and $A > 0$ such that $||\boldsymbol{u} \cdot \mathrm{P}_M^{(t)} - \boldsymbol{\pi}|| \leq Ab^t$. Our estimate of the mixing time for global convergence to the stationary distribution is $t_{\text{mix}} \sim H_M^{-1}$. such that the system converges to $\boldsymbol{\pi}$ exponentially in time given the entropy rate H_M of process M as

$$||\boldsymbol{u} \cdot \mathrm{P}_M^{(t)} - \boldsymbol{\pi}|| \sim e^{-H_M t}$$
$$\text{where } H_M = -\sum_{i,j \in S} \log(P(i,j)) P(i,j)\, \pi_j. \tag{15}$$

This convergence rate estimate for $s_i \in S$ is given by Shannon's entropy [32]. With the number of visits to a state $N_i : i \in S$ and a given $\epsilon \in (0,1)$ we have

$$\Pr[(1-\epsilon)\pi_i t < N_i < (1+\epsilon)\pi_i t] = 1 - e^{-\epsilon \pi_i h_i t}$$
$$\text{where } h_i = -\sum_{j \in S} \log(P(j,i)) P(j,i)$$

4.3 Chain Growth Properties

Definition 5. ***Characteristic string of*** M**.** Let $\mathtt{sl} = \{\mathtt{sl}_i : 0 < i \leq T\} \in \{\mathbb{N}_0\}^*$ be a sequence of slots where T is the number of time steps executed by M with an arbitrary input distribution $\boldsymbol{u}$, honest participants $\mathcal{H}$, adversary $\mathcal{A}$ and environment $\mathcal{Z}$. We define the characteristic string $w \in \Lambda^*$ as a mapping of $\mathtt{sl}$ onto a space of labels $w_i \in \Lambda$ where the labels map to configurations of the set of elected leaders $w : \mathcal{P}(\cdot), \{\mathbb{N}_0\} \to \Lambda^*$ as defined by the characteristic map:

$$w_i = \begin{cases} \perp & \textit{if } \mathcal{P}(\mathtt{sl}_i) = \emptyset \\ \mathcal{H}_0 & \textit{if } |\mathcal{P}(\mathtt{sl}_i)| = 1 \wedge \textit{ elected party is honest} \\ \mathcal{H}_1 & \textit{if } |\mathcal{P}(\mathtt{sl}_i)| > 1 \wedge \textit{ all elected parties are honest} \\ \mathcal{A}_1 & \textit{if } |\mathcal{P}(\mathtt{sl}_i)| > 0 \wedge \textit{ any elected party is adversarial} \end{cases} \tag{16}$$

Definition 6. ***Reduction maps.*** We define the maps $\rho_\Lambda : \Lambda^* \to \Lambda'^*$ and $\rho_{\Lambda'} : \Lambda'^* \to \{0,1\}^*$ building on Definition 1 with $i \in \Lambda$, $w \in \Lambda^*$, $j \in \Lambda'$, $v \in \Lambda'^*$:

$$\rho_\Lambda(i||w) = \begin{cases} \perp ||\rho_\Lambda(w) & \textit{if } i = \perp \\ 0 ||\rho_\Lambda(w) & \textit{if } i = \mathcal{H}_0 \\ 1 ||\rho_\Lambda(w) & \textit{Otherwise} \end{cases} \tag{17}$$

$$\rho_{\Lambda'}(j||v) = \begin{cases} \rho_{\Lambda'}(v) & \textit{if } j = \perp \\ j||\rho_{\Lambda'}(v) & \textit{Otherwise} \end{cases} \tag{18}$$

Forkable string representation. Using a random mapping representation (Appendix D), we now formally define the forkable string $z \in \{0,1\}^*$ as $z = \rho_{\Lambda'}(x)$ with $z_m \in \{0,1\}$ to yield normalized probabilities of forkable events:

$$p_0 = \Pr[z_m = 0] = \frac{\Pr[x_l = 0]}{\Pr[x_l = 0] + \Pr[x_l = 1]}$$
$$p_1 = \Pr[z_m = 1] = \frac{\Pr[x_l = 1]}{\Pr[x_l = 0] + \Pr[x_l = 1]} \tag{19}$$

From Equation 32, with α_p as the relative stake of p, it follows that

$$p_0 \geq \frac{\sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}))\phi(j,\alpha_{\mathcal{H}})}{1-\sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}+\alpha_{\mathcal{A}}))} \tag{20}$$

Let $g_\lambda(t) = \Pr[w_i = \lambda, t] : \lambda \in \Lambda, i \in \{t\}$. Define $g_{\mathcal{H}_0}(t)$ as the expected number of unique honest leaders in t slots with time-homogeneous growth rates such that

$$\begin{aligned} g_{\mathcal{H}_0}(t) = g_0 t : g_0 &= \sum_{j \in S} \pi_j P(j,1) \Pr[w_i = \mathcal{H}_0, j] \\ &\geq \sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}))\phi(j,\alpha_{\mathcal{H}}) \end{aligned} \tag{21}$$

Let $g_{\mathcal{A}_1}(t)$ be the expected number of adversarial leaders in t slots such that

$$\begin{aligned} g_{\mathcal{A}_1}(t) = g_1 t : g_1 &= \sum_{j \in S} \pi_j P(j,1) \Pr[w_i = \mathcal{A}_1, j] \\ \leq \sum_{j \in S} \pi_j &\Big(\phi(j,\alpha_{\mathcal{A}}) + \phi(\delta,\alpha_{\mathcal{H}}) - (1-\phi(j,\alpha_{\mathcal{H}}))\phi(j,\alpha_{\mathcal{H}}) \Big) \end{aligned} \tag{22}$$

and $g_{\not\perp}(t)$ be the expected number of all leaders in t slots, then

$$g_{\not\perp}(t) = g_{\not\perp} t : g_{\not\perp} = 1 - \sum_{j \in S} \pi_j \big(1-\phi(j,\alpha_{\mathcal{H}}+\alpha_{\mathcal{A}})\big). \tag{23}$$

We evaluate the concentration bounds based on the Chernoff-Hoeffding theorem [11] in Theorem D in the appendix.

4.4 Taktikos Security Properties

Now we may evaluate the security properties of the protocol Π^{Tak} in terms of common prefix, chain growth, and chain quality from [13].

Theorem 2 (Common prefix). Let $k, T \in \mathbb{N}$ and $a_u > 0$. When executed in a synchronous environment in T slots according to process M, the probability that Π^{Tak} violates the common prefix property for some $0 < \zeta < 1$ is no greater than

$$e^{\ln(T)-\Omega(k)} + a_u u_\pi e^{-H_\epsilon \zeta^2 \left(\frac{g_1 T}{1-\zeta} - \frac{g_{\not\perp} T}{1+\zeta}\right)} + e^{-H_M T}$$

if there exists $\epsilon \in (0,1)$ such that

$$\frac{\sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}))\phi(j,\alpha_{\mathcal{H}})}{\sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}+\alpha_{\mathcal{A}}))} \geq \frac{1}{2}(1+\epsilon)$$

Theorem 3 (Chain growth). Let $T \in \mathbb{N}$. When executed in a synchronous environment in T slots according to process M, the probability that Π^{Tak} violates the chain growth property for some $0 < \zeta < 1$ with chain growth rate $g_0 = \sum_{j \in S} \pi_j (1-\phi(j,\alpha_{\mathcal{H}}))\phi(j,\alpha_{\mathcal{H}})$ is no greater than

$$a_u u_\pi e^{-H_\epsilon \zeta^2 \frac{g_0 T}{1+\zeta}} + e^{-H_M T}$$

Theorem 4 (Chain quality). Let $k, T \in \mathbb{N}$. When executed in a synchronous environment in T slots according to process M, the probability that Π^{Tak} violates the chain quality property for some $0 < \zeta < 1$ with chain quality coefficient

$$1 - \frac{g_1}{g_{\not\perp}} \leq e^{\ln(T) - \Omega(k)} + a_u u_\pi e^{-H_\epsilon \zeta^2 \left(\frac{g_0 T}{1+\zeta} + \frac{g_1 T}{1-\zeta} + \frac{g_{\not\perp} T}{1+\zeta^2}\right)} + e^{-H_M T}$$

if there exists $\epsilon \in (0, 1)$ such that

$$\frac{\sum_{j \in S} \pi_j (1 - \phi(j, \alpha_{\mathcal{H}})) \phi(j, \alpha_{\mathcal{H}})}{\sum_{j \in S} \pi_j (1 - \phi(j, \alpha_{\mathcal{H}} + \alpha_{\mathcal{A}}))} \geq \frac{1}{2}(1 + \epsilon)$$

5 Related Work

Ouroboros [22] forms the foundation of Nakamoto-style probabilistic proof of stake and presents a formal framework for reasoning about blockchain consistency. Subsequent variants [3,4,13] extend the protocol to new security settings, improving the original approach and incorporating new formal techniques to tighten security bounds. However, these approaches do not consider LDD and require adaptation to our domain. Similarly, our assessment of grinding extends previous work [8,21], but reframes the discussion to accommodate our modifications.

The use of time-dependent agreement for multi-agent consensus appears in other domains previously, such as control theory [26]. These ideas contain similarities to the concept of LDD as presented in this work. However, previous approaches do not consider Byzantine faults, making them impractical for adaptation to the distributed ledger domain.

The use of Markov chains to analyze the properties of consensus mechanisms appears frequently in the literature, including in [22], where consistency properties arise from other techniques and do not directly apply under the conditional probabilities that emerge from LDD. We stress that our approach uses a completely different state space, corresponding to the discrete values of the slot interval. Our approach borrows inspiration from the framework proposed in [23], which uses Markov chains to evaluate consistency from the perspective of PoW. However, our approach extends the idea to allow for more robust representations and in particular, local dynamic difficulty variations.

6 Conclusion

In this work, we present Ouroboros Taktikos, a proof of stake protocol that improves the performance of Nakamoto-style probabilistic consensus. This work introduces the concept of conditional eligibility testing, termed local dynamic difficulty, and shows that adopting a time-varying difficulty curve offers advantages over the static threshold test. We implement this protocol and empirically evaluate its performance against state-of-the-art, finding a $\sim 3\times$ improvement in block throughput and a $\sim 5\times$ reduction in 99^{th} percentile block latency. Finally, we develop and present a novel discrete-time Markov model for analyzing the dynamics and steady-state behavior of stochastic eligibility computations in our setting, representing an optimal adversary in the (semi-)synchronous setting. The robust agreement between empirical evaluation and theoretical prediction, replicating observed real-world behavior, supports the security and throughput improvements in Taktikos and adds the powerful tool of time-based regularization to the eventual consensus toolbox.

References

1. Input-output-hk/cardano-node: The core component that is used to participate in a Cardano decentralised blockchain. https://github.com/input-output-hk/cardano-node (2022)
2. Algoet, P.H., Cover, T.M.: A Sandwich Proof of the Shannon-McMillan-Breiman Theorem. The Annals of Probability **16**(2), 899–909 (1988)
3. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 913–930. CCS '18, Association for Computing Machinery, New York, NY, USA (Oct 2018). https://doi.org/10.1145/3243734.3243848
4. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros Chronos: Permissionless Clock Synchronization via Proof-of-Stake. Tech. Rep. 838 (2019)
5. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Dynamic Ad Hoc Clock Synchronization. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 399–428. Lecture Notes in Computer Science, Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_14
6. Bagaria, V., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Proof-of-Stake Longest Chain Protocols: Security vs Predictability (Feb 2020). https://doi.org/10.48550/arXiv.1910.02218
7. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. pp. 431–448. CRYPTO '99, Springer-Verlag, Berlin, Heidelberg (Aug 1999)
8. Blum, E., Kiayias, A., Moore, C., Quader, S., Russell, A.: The Combinatorics of the Longest-Chain Rule : Linear Consistency for Proof-of-Stake Blockchains (2019)
9. Buterin, V., Griffith, V.: Casper the Friendly Finality Gadget. arXiv:1710.09437 [cs] (Jan 2019)
10. Canetti, R.: Universally composable signature, certification, and authentication. In: Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. pp. 219–233 (Jun 2004). https://doi.org/10.1109/CSFW.2004.1310743
11. Chung, K.M., Lam, H., Liu, Z., Mitzenmacher, M.: Chernoff-Hoeffding Bounds for Markov Chains: Generalized and Simplified. In: Dürr, C., Wilke, T. (eds.) 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012). Leibniz International Proceedings in Informatics (LIPIcs), vol. 14, pp. 124–135. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2012). https://doi.org/10.4230/LIPIcs.STACS.2012.124
12. Daian, P., Pass, R., Shi, E.: Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security. pp. 23–41. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-32101-7_2
13. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 66–98. Lecture Notes in Computer Science, Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_3

14. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a Race and Nakamoto Always Wins. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 859–878. ACM, Virtual Event USA (Oct 2020). https://doi.org/10.1145/3372297.3417290
15. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) Public Key Cryptography - PKC 2005. pp. 416–431. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30580-4_28
16. Fan, L., Zhou, H.S.: A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto's Design via Proof-of-Stake). Cryptology ePrint Archive (2017)
17. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin Backbone Protocol: Analysis and Applications. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. pp. 281–310. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
18. Gaži, P., Kiayias, A., Russell, A.: Tight Consistency Bounds for Bitcoin. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 819–838. Association for Computing Machinery, New York, NY, USA (Oct 2020)
19. 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. pp. 51–68. SOSP '17, Association for Computing Machinery, New York, NY, USA (Oct 2017). https://doi.org/10.1145/3132747.3132757
20. Kiayias, A., Panagiotakos, G.: Speed-Security Tradeoffs in Blockchain Protocols. Cryptology ePrint Archive (2015)
21. Kiayias, A., Quader, S., Russell, A.: Consistency of Proof-of-Stake Blockchains with Concurrent Honest Slot Leaders. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). pp. 776–786 (Nov 2020). https://doi.org/10.1109/ICDCS47774.2020.00065
22. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 357–388. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
23. Kiffer, L., Rajaraman, R., shelat, a.: A Better Method to Analyze Blockchain Consistency. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 729–744. CCS '18, Association for Computing Machinery, New York, NY, USA (Oct 2018). https://doi.org/10.1145/3243734.3243814
24. Küfeoğlu, S., Özkuran, M.: Bitcoin mining: A global review of energy and power demand. Energy Research & Social Science **58**, 101273 (Dec 2019). https://doi.org/10.1016/j.erss.2019.101273
25. Li, W., Andreina, S., Bohli, J.M., Karame, G.: Securing Proof-of-Stake Blockchain Protocols. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) Data Privacy Management, Cryptocurrencies and Blockchain Technology. pp. 297–315. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_17
26. Lorenz, J., Lorenz, D.A.: On Conditions for Convergence to Consensus. IEEE Transactions on Automatic Control **55**(7), 1651–1656 (Jul 2010). https://doi.org/10.1109/TAC.2010.2046086

27. Meyn, S.P., Tweedie, R.L.: Markov Chains and Stochastic Stability. Springer Science & Business Media (Dec 2012)
28. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Tech. rep. (2008)
29. Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). pp. 358–367. IEEE, Toulouse (Jul 2016). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0070
30. O'Dwyer, K.J., Malone, D.: Bitcoin Mining and its Energy Footprint pp. 280–285 (Jan 2014). https://doi.org/10.1049/cp.2014.0699
31. Pass, R., Seeman, L., Shelat, A.: Analysis of the Blockchain Protocol in Asynchronous Networks. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 643–673. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
32. Rhodes, J., Schilling, A.: Bounds on mixing time of finite Markov chains (Oct 2020). https://doi.org/10.48550/arXiv.2010.08879
33. Ullrich, J., Stifter, N., Judmayer, A., Dabrowski, A., Weippl, E.: Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids. In: Bailey, M., Holz, T., Stamatogiannakis, M., Ioannidis, S. (eds.) Research in Attacks, Intrusions, and Defenses. pp. 184–203. Lecture Notes in Computer Science, Springer International Publishing (2018)

Appendix

Blockchain Preliminaries

A *blockchain* $\mathcal{C}$ consists of a sequence of ℓ concatenated blocks $B_0\|B_1\|B_2\|\cdots\|B_\ell$, where $\ell \geq 0$ and B_0 is the initial block (genesis block). If a chain $\mathcal{C}$ is truncated the last λ blocks, we write $\mathcal{C}^{\lceil\lambda}$. The length of a chain $\mathsf{len}(\mathcal{C})$ is the number of blocks. The block B_ℓ is the head of the chain, denoted $\mathsf{head}(\mathcal{C})$. We let $\mathcal{C}_1 \preceq \mathcal{C}_2$ indicate that the chain $\mathcal{C}_1$ is a prefix of the chain $\mathcal{C}_2$.

We refer to the following definitions from [3,13]. In the static stake case, we also assume that a fixed collection of n stakeholders $\mathcal{P} = (U_1, \cdots, U_n)$ interact throughout the protocol. Stakeholder U_i has stake σ_i at the beginning of the protocol.

Definition 7 ([13]). ***Genesis Block.*** *The genesis block B_0 contains the list of stakeholders identified by a label U_i, their respective public keys and respective stakes*

$$\mathbb{S}_0 = \Big((U_1, v_1^{\text{vrf}}, v_1^{\text{kes}}, v_1^{\text{dsig}}, \sigma_1), \cdots, (U_n, v_n^{\text{vrf}}, v_n^{\text{kes}}, v_n^{\text{dsig}}, \sigma_n)\Big)$$

and a nonce $\eta \in \{0,1\}^k$.

Definition 8. ***Relative Stake.*** *Let $\mathcal{P}$ be the set of all stakeholders and total stake $\Sigma = \sum_{U_i \in \mathcal{P}} \sigma_i$, then a relative stake of a party $U_i \in \mathbb{P}$ is $\alpha_i = \sigma_i/\Sigma$. Let $\sigma_{\mathcal{H}}$, $\sigma_{\mathcal{A}}$ be the total stake of honest stakeholders and the adversary, respectively. Correspondingly $\alpha_{\mathcal{H}} = \sigma_{\mathcal{H}}/\Sigma$ and $\alpha_{\mathcal{A}} = \sigma_{\mathcal{A}}/\Sigma$.*

Security Properties

Previously, several fundamental security properties for PoW protocols have been defined: *common prefix property* [17,31], *chain quality property* [17], and *chain growth property* [20]. A blockchain protocol is secure if it satisfies the following properties:

Definition 9 (Common Prefix Property with parameter $k \in \mathbb{N}$). *The chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest parties at the onset of the slots $\mathtt{sl}_1 < \mathtt{sl}_2$ are such that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.*

Definition 10 (Chain Quality Property with parameters $k \in \mathbb{N}$ and $\mu \in (0, 1]$). *Consider any portion of length at least k of the chain possessed by an honest party at the onset of a round; the ratio of blocks originating from the adversary is at most $1 - \mu$. We call μ the chain quality coefficient.*

Definition 11 (Chain Growth Property with parameters g). *For $g \in (0, 1], s \in \mathbb{N}$, consider the chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest parties at the onset of the slots $\mathtt{sl}_1, \mathtt{sl}_2$ with $\mathtt{sl}_2$ at least s slots ahead of $\mathtt{sl}_1$. Then it holds that $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g \cdot s$. We call g the speed coefficient.*

Ideal Functionalities

In this section, we introduce the ideal functionalities $\mathcal{F}_{\mathsf{INIT}}$, $\mathcal{F}_{\mathsf{VRF}}$, $\mathcal{F}_{\mathsf{KES}}$, $\mathcal{F}_{\mathsf{DSIG}}$ required by our protocol in the hybrid model.

Ideal signature scheme $\mathcal{F}_{\mathsf{DSIG}}$. The definition of ideal signature scheme functionality $\mathcal{F}_{\mathsf{DSIG}}$ was first formalized in [10] We refer to the details of $\mathcal{F}_{\mathsf{DSIG}}$ in [10].

Ideal key evolving signature $\mathcal{F}_{\mathsf{KES}}$. In regular digital signatures, if the signing key is compromised, all signatures, even those that were issued by the honest signer before the compromise, will not be trustworthy any more. The notion of forward secure signatures [7] was formally introduced to address this major shortcoming. They are digital signature schemes with the property that compromising the signing key of a user at a certain time period does not allow for the forgery of signatures from the past. In 2018, David et al. [13] was the first to define an ideal functionality $\mathcal{F}_{\mathsf{KES}}$ for forward secure signature schemes. $\mathcal{F}_{\mathsf{KES}}$ is distinguished from $\mathcal{F}_{\mathsf{DSIG}}$ because $\mathcal{F}_{\mathsf{KES}}$ only allows forgery of a corrupted signer with the current and future signing keys. This implies that the security of a key-evolving signature generated before the compromise is still maintained. In more detail, the functionality $\mathcal{F}_{\mathsf{KES}}$ has three phases: key generation, sign-update, and signature verification. In the key generation phase, each signer with a verification v has a corresponding counter to keep track the total number of signature updates. In the sign-update phase, the functionality only allows signing if the total number of signatures signed with the current key is less than a maximum value called the key time step. This step models the key-evolving operation in forward secure signature schemes. In the signature verification phase, the requirements in $\mathcal{F}_{\mathsf{DSIG}}$

and $\mathcal{F}_{\mathsf{KES}}$ only lets the adversary set the response to a verification query if the key has not been updated. We refer to the details of $\mathcal{F}_{\mathsf{KES}}$ in [13].

Ideal verifiable random function $\mathcal{F}_{\mathsf{VRF}}$. In the standard verifiable random function scheme (VRFs) [15], the adversary is able to see and skew the output distribution of the VRF. Therefore, to ensure that the adversary cannot influence the outputs to obtain more advantage than honest stakeholders in the leader election process, a strong security requirement is necessary for these VRFs. David et al. [13] formalized the ideal verifiable random function $\mathcal{F}_{\mathsf{VRF}}$ with an additional property called *unpredictability under malicious key generation*. The functionality $\mathcal{F}_{\mathsf{VRF}}$ proceeds as follows: upon receiving the key generation message of a stakeholder U_i, the ideal functionality pairs the verification key from the ideal adversary and records the pair. For VRF evaluation of input m from the stakeholder, the functionality will sample a random output and associate it with input m. If the stakeholder asks for VRF evaluation and proof, a VRF proof π will be chosen (from the ideal adversary) and recorded with the input m and the VRF output. Note that the unpredictability is captured by constraining any influence of the ideal adversary on the random output. We refer to [13] for further details of this functionality.

Ideal functionality $\mathcal{F}_{\mathsf{INIT}}$. The new stake distribution and random nonce for leader election would normally be updated every epoch but in this setting we consider static stake and a constant nonce given by the genesis block. This functionality will later be instantiated by a real protocol in the dynamic stake setting. $\mathcal{F}_{\mathsf{INIT}}$ will record all the stakes with their corresponding verification keys $\mathbb{S}$, then distributes it along with a random nonce η to registered stakeholders in the system. We refer to the details of $\mathcal{F}_{\mathsf{INIT}}$ in [13].

We adopt the security model from [13] with an ideal functionality $\mathcal{F}$. This includes *a diffuse functionality* and *a key and transaction functionality* as described in [22]. A real-world protocol Π enhanced with an ideal functionality $\mathcal{F}$ is executed in a synchronous network model that involves the following components:

Stakeholders $U_1, \cdots, U_n$. Each stakeholder maintains a local state and a local chain $\mathcal{C}_{\mathtt{loc}}$. In each slot, each stakeholder tries to extend the chain by attempting to build a new block at the end of their chain.

Adversary $\mathcal{A}$ and network delivery. We consider a setting where time is divided into slots. To simplify our Markov chain model and analysis, we consider a synchronous setting where each party has access to a *synchronized* clock. In this model, the adversary $\mathcal{A}$ is responsible for delivering messages between stakeholders, but not allowed to change the content of the message or prevent them from being delivered via a *diffuse* functionality.

Environment $\mathcal{Z}(1^k)$. At each slot $\mathtt{sl}$, the environment $\mathcal{Z}$ is allowed to activate any subset of stakeholders it wishes. Each one of them will possibly produce messages that are to be transmitted to other stakeholders.

Random Oracle H. All stakeholders have access to a random oracle $H : \{0,1\}^* \to \{0,1\}^k$. This oracle will deterministically answer every input query from a stakeholder with an independent and uniformly random string from $\{0,1\}^k$.

We now describe an execution of a blockchain protocol Π given the above components. At the beginning, the environment $\mathcal{Z}(1^k)$ instantiates n stakeholders $U_1, \cdots U_n$. The protocol proceeds in slots as follows. At each slot, each player U_i does the following: (i) U_i receives blocks created by other stakeholders and includes the blocks in its chain based on the protocol Π; (ii) U_i receives some data from $\mathcal{Z}(1^k)$ and attempts to create a new block which includes the data from $\mathcal{Z}$; and (iii) if U_i succeeds in generating a new valid block, then U_i sends the block to other players via a diffuse functionality. Note that the adversary can only corrupt a stakeholder if it is given permission by the environment $\mathcal{Z}$ through a message $(\mathsf{Corrupt}, U)$. Similar to the model in [22], the environment $\mathcal{Z}$ can give arbitrary permissions, however with respect to an initial stakeholder set and respective stake, the adversary will be restricted to controlling only a percentage of that stake α.

A Bounded Delay Model

The techniques used in this work are general to any difficulty curve so the underlying protocol of Praos represents a choice of $f(\delta) = f_B$. This also corresponds to the staking procedure of Genesis but that security setting is time-inhomogeneous so we focus on Praos. Figure 5 shows the Markov process diagram with $\gamma = \Delta - 1$ and constant f. The reduction map for the bounded network delay $\Delta \in \mathbb{N}_0$ is included here, modeled after [13]. This corresponds to associating each event with the Δ-right isolated condition that accommodates the time for nodes to synchronize.

Our environment is assumed synchronous, so we only survey the proportion of forkable events given the Δ-right isolation rule (honest events $\mathcal{H}_0$ must be followed by $\Delta - 1$ empty slots, i.e. $\mathcal{H}_0 || \{\perp\}^{\Delta-1} \to 0$, and anything else is deemed forkable). The reduction map $\rho_\Delta : L^* \to L'^*$ assuming the Δ-right isolation rule is

$$\rho_\Delta(i||w) = \begin{cases} \rho_\Delta(w) & \text{if } i = \perp \\ 0||\rho_\Delta(w') & \text{if } i = \mathcal{H}_0 \wedge w = \{\perp\}^{\Delta-1}||w' \\ 1||\rho_\Delta(w) & \text{if } i = \mathcal{H}_0 \vee i = \mathcal{H}_1 \vee i = \mathcal{A}_1 \end{cases} \tag{24}$$

For the bounded delay model we set $\gamma = \Delta - 1$. Let $x = \rho_\Delta(w) \in L'^*$. For $x_l \in L'$ drawn from x we have

$$\begin{aligned} \Pr[x_l = 0] &= \sum_{\Delta \leq j \in S} \pi_j P(j,1) \Pr[w_i = \mathcal{H}_0, j] \\ \Pr[x_l = \perp] &= \pi_\Delta P(\Delta, \Delta) \Pr[w_i = \perp, \Delta] \\ &+ \sum_{j < \Delta \in S} \pi_j P(j, j+1) \Pr[w_i = \perp, j] \\ \Pr[x_l = 1] &= \sum_{\Delta > j \in S} \pi_j P(j,1) \Pr[w_i = \not\perp, j] \\ &+ \sum_{\Delta \leq j \in S} \pi_j P(j,1) \big(\Pr[w_i = \mathcal{H}_1, j] + \Pr[w_i = \mathcal{A}_1, j] \big) \end{aligned} \tag{25}$$

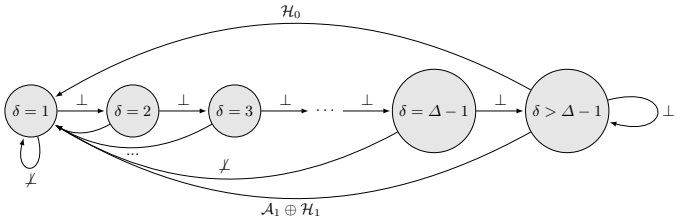

Fig. 5: The Markov process diagram for $\gamma = \Delta - 1$. This reproduces the security properties of Ouroboros Praos with the Δ-right isolation rule given by ρ_Δ. Refer to [$**$] for an interactive plot where Δ and f_B are adjustable parameters. The security properties of Praos may be evaluated by setting $\Delta > 0$, $\gamma = 0$, and $f = f_B > 0$ in the interactive plots.

A.1 Chain Selection Rule

We propose a `maxvalid-tk` chain selection rule, a method to address the semi-synchronous setting by adapting the `maxvalid-mc` and `maxvalid-bg` selection rules from Praos and Genesis [13,3]. We build on the `maxvalid-mc` rule since our environment is statically registered. For the set of chains collected from the network $\{\mathcal{C}_1, ..., \mathcal{C}_j\}$, security checkpoint depth $k \in \mathbb{N}$, and local chain $\mathcal{C}_{\texttt{loc}}$ let the chain selection algorithm be defined by Algorithm 1. This simple modification would transform Δ-divergences to unique convergence opportunities. The algorithm `maxvaild-slot`$(\mathcal{C})$ returns the maximum slot, i.e. the slot of the head of the chain $\mathcal{C}$. The earliest slot after each block is biased towards the honest majority with an appropriately chosen Taktikos difficulty curve. To see this, one may explore the proportion of unique convergence events, honest ties, and adversarial leader elections in the Taktikos curve parameter space using [$**$] that includes the bounded delay Δ.

B Strong Ergodicity

The Taktikos curve forging window introduces a different regime of stability, in the sense that the local difficulty change tends to keep the system in the state space of S_γ for a given $f(\delta)$ with sufficient parameters and resources. We have chosen a curve that induces geometric ergodicity on the subset of states S_γ, that can be relaxed to couple with the recovery phase. The recovery phase state $s_{\delta>\gamma}$ corresponds to exponential ergodicity because its distribution converges exponentially as $\delta \to \infty$. As a consequence the two systems act as coupled Markov chains that exhibit strong ergodicity. This sustained and enhanced stability is maintained by the mutual interaction of staking parties reacting to the most recent blocks on the network. Consider the system isolated to S_γ. The stationary distribution has the form

$$\pi(\delta, r) = C \prod_{i}^{\delta-1} (1 - ci)^r \sim e^{r - r\delta} (1 - c\delta + c)^{r\delta - \frac{r}{c} - r} \tag{26}$$

Algorithm 1: `maxvalid-tk` $(k, \mathcal{C}_{\texttt{loc}}, \mathcal{C}_1, ..., \mathcal{C}_j)$

```
C ← C_loc;
for i ∈ {1, ..., j};
 do
    if IsValidChain(C_i);
     then
        if C_i forks from C at most k blocks;
         then
            if |C_i| = |C|;
             then
                sl_i ← maxvalid-slot(C_i);
                sl_l ← maxvalid-slot(C);
                if sl_i < sl_l;
                 then
                    C ← C_i;
                if |C_i| > |C|;
                 then
                    C ← C_i;
return C
```

and we show that this distribution converges geometrically at the point δ_c

$$\pi(\delta_c, r) = (1 - c(\delta_c - 1))^r \pi(\delta_c - 1, r) = 0. \tag{27}$$

For $\pi(\delta_c - 1) > 0$ this gives

$$\delta_c = 1 + \frac{1}{c} \tag{28}$$

where $\pi(\delta \geq \delta_c) = 0$.

B.1 Entropy Rate Optimization

Here we speculate that the linear curve may be relaxed to give a unique slope that optimizes the mixing time of the forging window. The linear profile was chosen for simplicity of security analysis but any general curve can be constructed and evaluated in our security framework.

Theorem 5. ***Entropy Rate Optimization.*** For $f(\delta)$ where δ is the γ bounded parameter corresponding to process M, assume that $f(\delta)$ increases monotonically over the domain $0 < \delta \leq \gamma$ and $f(\gamma+1)$ is a fixed constant where $f(\gamma) > f(\gamma+1)$. With $r = \frac{1}{2}$, the optimal curve is given by the local extreme of the entropy rate with appropriately constrained $f(\delta)$ such that

$$\partial_{f(\delta)} H = 0 \tag{29}$$

where ergodicity of process M guarantees that $H|_{f(\delta)}$ is at a local maximum and $f'(\delta) = \frac{df(\delta)}{d\delta} > 0$ over $0 < \delta \leq \gamma$ where

$$\lim_{\delta \to 0} f(\delta) = 0,\ f'(0) = 0.$$

C Protocol Details

Algorithm 2: GrindingSim1 : $\mathcal{Y}, r, f \to [0,1]$

Require : $y_i \in [0,1]$ for $y_i \in \mathcal{Y}$, $r \in [0,1]$, $f : \mathbb{N} \to [0,1]$
$L \leftarrow |\mathcal{Y}|$;
$\ell_{\max} \leftarrow 0$;
$B \leftarrow [(0,0)]$;
for $y_i \in \mathcal{Y}$;
do
  $B' \leftarrow [\,]$;
  for $b \in B$;
  do
    $(s, \ell) \leftarrow b$;
    if $y_i < 1 - \big(1 - f(i-s)\big)^r$;
    then
      $B' \leftarrow (i, \ell + 1) || B'$;
  $B \leftarrow B' || B$;
  for $b \in B$ **do**
    $(s, \ell) \leftarrow b$;
  $\ell_{\max} \leftarrow \mathsf{max}(\ell, \ell_{\max})$;
return $\ell_{\max}/L$

C.1 Consistency Bound Under Grinding

Direct comparison between consistency bounds of protocols poses a challenge due to differences between assumptions around both design and operation. In Figure 1(a), we illustrate this bound in terms of block proposals per delay interval, which varies based on block frequency and network delay.

To illustrate this challenge, we will assume for the sake of example that all protocols assume a network delay of five seconds. Similarly, we will take the mean values from each protocol as the canonical block frequencies: 6.67 seconds for Taktikos, and 19.29 seconds for Praos. Therefore, the ratio of blocks per delay interval becomes $\frac{5}{19.29} = 0.259$ for Praos and $\frac{5}{6.67} = 0.750$ for Taktikos.

When plotting these comparable values on Figure 1(b), we find that Taktikos shows a consistency bound of 0.451 while Praos shows a value of 0.463. These values represent a relative difference of 2.6%; that is, Taktikos achieves 97.4% of Praos' consistency bound while substantially improving performance (as detailed in Section 3). However, since the protocols do not evaluate at the same point on the X axis, these results may be initially counter-intuitive from the visual plot.

Note that the chosen network delay and block frequency values were selected for convenience of comparison. In practice, these values will vary based on network

Algorithm 3: $\mathsf{GrindingSim2} : \mathcal{Y}, r, d, f \to [0,1]$

Require : $y_i \in [0,1]$ for $y_i \in \mathcal{Y}$, $r \in [0,1]$, $d \in \mathbb{N}_1$, $f : \mathbb{N} \to [0,1]$;
$L \leftarrow |\mathcal{Y}|$;
$\ell_{\max} \leftarrow 0$;
$B \leftarrow [(0,0)]$;
for $y_i \in \mathcal{Y}$;
do
  $B' \leftarrow [\,]$;
  for $b \in B$;
  do
    $(s, \ell) \leftarrow b$;
    if $y_i < 1 - \big(1 - f(i-s)\big)^r$;
    then
      $B' \leftarrow (i, \ell+1)||B'$;
  $B \leftarrow B'||B$;
  for $b \in B$;
  do
    $(s, \ell) \leftarrow b$;
    $\ell_{\max} \leftarrow \mathsf{max}(\ell, \ell_{\max})$;
  $B' \leftarrow [\,]$;
  for $b \in B$;
  do
    $(s, \ell) \leftarrow b$;
    if $\ell_{\max} - \ell < d$;
    then
      $B' \leftarrow b||B'$;
  $B \leftarrow B'$;
return $\ell_{\max}/L$;

conditions and stochastic variability, so fixed-point comparisons provide only a small window into behavioral differences. As network delay (the numerator) decreases, these points approach 0 and the differences in consistency bound also decrease.

Finally, we provide an intuition behind the asymptotic behavior as d increases. While the adversary may track many possible tines, the value of those tines is a function of (1) random chance and (2) honest behavior. Each held-back tine becomes useless if either it naturally succumbs to stochastic variability – it falls too far behind the tip – or if an honest node claims that slot's unused eligibility. Thus, even when tracking a very large number of deep tines, nearly all must be discarded and the small fraction remaining provide the adversary only a modest grinding advantage.

D Framework Supplementals

Random mapping representation of M. Let the random mapping representation $R_\Lambda : S, \Lambda \to S$ of matrix P_M be defined in terms of the conditional variable $w_i \in \Lambda$ and $j, k \in S$ where x_l is drawn from $x = \rho_\Lambda(w) \in \Lambda'^*$ and $w \in \Lambda^*$. The probabilities of each path being taken are defined by

$$\Pr[R_\Lambda(j, w_i) = k] = P(j, k) \to \Pr[w_i, \delta] P(\delta, k) = p_{w_i}(\delta) \tag{30}$$

Let the resource factor r correspond to the net active stake distributed across $\mathcal{H}$ and $\mathcal{A}$ such that $r = \alpha_\mathcal{H} + \alpha_\mathcal{A}$. The probabilities for each label in Λ are given by

$$\begin{aligned}
&\Pr[w_i = \perp, \delta] P(\delta, \delta + 1) = p_\perp(\delta) = 1 - \phi(\delta, \alpha_\mathcal{H} + \alpha_\mathcal{A}) \\
&\Pr[w_i = \mathcal{A}_1, \delta] P(\delta, 1) = p_{\mathcal{A}_1}(\delta) = \phi(\delta, \alpha_\mathcal{A}) \\
&\Pr[w_i = \mathcal{H}_0, \delta] P(\delta, 1) = p_{\mathcal{H}_0}(\delta) \\
&\quad \geq (1 - \phi(\delta, \alpha_\mathcal{H}))\phi(\delta, \alpha_\mathcal{H}) \\
&\Pr[w_i = \mathcal{H}_1, \delta] P(\delta, 1) = p_{\mathcal{H}_1}(\delta) = \phi(\delta, \alpha_\mathcal{H}) - p_{\mathcal{H}_0}(\delta) \\
&\quad \leq \phi(\delta, \alpha_\mathcal{H}) - (1 - \phi(\delta, \alpha_\mathcal{H}))\phi(\delta, \alpha_\mathcal{H})
\end{aligned} \tag{31}$$

where $\phi(\delta, \alpha) = 1 - \big(1 - f(\delta)\big)^{\alpha}$. It follows that for $x_l \in L'$ we have

$$\begin{aligned}
\Pr[x_l = 0] &= \sum_{j \in S} \pi_j P(j,1) \Pr[w_i = \mathcal{H}_0, j] \\
&\geq \sum_{j \in S} \pi_j (1 - \phi(j, \alpha_{\mathcal{H}})) \phi(j, \alpha_{\mathcal{H}}) \\
\Pr[x_l = 1] &= \sum_{j \in S} \pi_j P(j,1) (\Pr[w_i = \mathcal{H}_1, j] + \Pr[w_i = \mathcal{A}_1, j]) \\
&\leq \sum_{j \in S} \pi_j (\phi(j, \alpha_{\mathcal{A}}) + \phi(j, \alpha_{\mathcal{H}}) - (1 - \phi(j, \alpha_{\mathcal{H}})) \phi(j, \alpha_{\mathcal{H}})) \\
\Pr[x_l = \perp] &= \pi_{\gamma+1} P(\gamma+1, \gamma+1) \Pr[w_i = \perp, \gamma+1] \\
&\quad + \sum_{j \in S_\gamma} \pi_j P(j, j+1) \Pr[w_i = \perp, j] \\
&= \sum_{j \in S} \pi_j (1 - \phi(j, \alpha_{\mathcal{H}} + \alpha_{\mathcal{A}}))
\end{aligned} \tag{32}$$

Theorem 6. ***Chernoff-Hoeffding Bounds.*** Let M be an ergodic discrete-time Markov chain with state space S and stationary distribution $\boldsymbol{\pi}$. Let $\mathcal{T}$ be the ϵ-mixing time for $\epsilon \leq 1/8$. Let $(\xi_1, \cdots, \xi_t)$ denote a t-step random walk on M starting from an initial distribution $\boldsymbol{u}$ on S, i.r., $\xi_1 \leftarrow \boldsymbol{u}$. For every $i \in \{t\}$, let $v_i : S \to [0,1]$ be a weight function at step i such that the expected weight $E_{\boldsymbol{v} \leftarrow \boldsymbol{\pi}}[v_i(\boldsymbol{v})] = g$ for all i. Define the total weight of the walk $(\xi_1, \cdots, \xi_t)$ by $\Xi = \sum_{i=1}^{t} v_i(\xi_i)$. There exists some constant $a_u > 0$ (which is independent of g, ζ, and ϵ) such that

$$\begin{aligned}
&\Pr[\Xi \geq (1+\zeta) g t] \leq a_u u_\pi \exp(-\zeta H_\epsilon g t) \text{ for } \zeta > 1 \\
&\Pr[\Xi \geq (1+\zeta) g t] \leq a_u u_\pi \exp(-\zeta^2 H_\epsilon g t) \text{ for } 0 \leq \zeta \leq 1 \\
&\Pr[\Xi \leq (1-\zeta) g t] \leq a_u u_\pi \exp(-\zeta^2 H_\epsilon g t) \text{ for } 0 \leq \zeta \leq 1
\end{aligned} \tag{33}$$

where u_π is the $\boldsymbol{\pi}$-norm of $\boldsymbol{u}$ given by $\sum_{i \in S} {u_i}^2 / \pi_i$ and $H_\epsilon = \frac{1}{72\mathcal{T}}$. Let $\zeta \in (0,1)$. The proportion of blocks originating from the adversary at time step T is at least $1 - \mu = \frac{g_1}{g_\not}$ with probability at most

$$1 - a_u u_\pi e^{-H_\epsilon \zeta^2 \left(\frac{g_1 T}{1-\zeta} - \frac{g_\not T}{1+\zeta} \right)}. \tag{34}$$

The ratio of unique honest blocks originating from honest stakeholders is at least $\mu = \frac{g_0}{g_\not}$ with probability at least

$$1 - a_u u_\pi e^{-H_\epsilon \zeta^2 \left(\frac{g_0 T}{1+\zeta} - \frac{g_\not T}{1-\zeta} \right)}. \tag{35}$$

$$\begin{bmatrix}
1 + b(1)^r & b(2)^r & b(3)^r & \ldots & b(\gamma)^r & b(\gamma+1)^r & 1 \\
-b(1)^r & 1 & 0 & \ldots & 0 & 0 & 0 \\
0 & -b(2)^r & 1 & \ldots & 0 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & 1 & 0 & 0 \\
0 & 0 & 0 & \ldots & -b(\gamma)^r & 1 - b(\gamma+1)^r & 0
\end{bmatrix} \tag{36}$$

Protocol $\Pi^{\mathbf{Tak}}$

The protocol Π^{Tak} runs by stakeholder $U_1, \cdots, U_n$ and ideal functionalities $\mathcal{F}_{\mathsf{INIT}}, \mathcal{F}_{\mathsf{VRF}}, \mathcal{F}_{\mathsf{KES}}, \mathcal{F}_{\mathsf{DSIG}}$, and random oracle H.

1. **Initialization**
   (a) The stakeholder U_i sends $(\mathsf{KeyGen}, sid, U_i)$ to the ideal functionalities $\mathcal{F}_{\mathsf{VRF}}, \mathcal{F}_{\mathsf{KES}}, \mathcal{F}_{\mathsf{DSIG}}$ and then receives $v_i^{\mathrm{vrf}}, v_i^{\mathrm{kes}}, v_i^{\mathrm{dsig}}$ from the ideal functionalities respectively.
   (b) The stakeholder U_i then registers $v_i^{\mathrm{vrf}}, v_i^{\mathrm{kes}}, v_i^{\mathrm{dsig}}$ to the functionality $\mathcal{F}_{\mathsf{INIT}}$ via command $(\mathsf{ver\text{-}keys}, sid, v_i^{\mathrm{vrf}}, v_i^{\mathrm{kes}}, v_i^{\mathrm{dsig}})$
   (c) The stakeholder U_i in the next rounds sends $(\mathsf{genblock\text{-}req}, sid, U_i)$ to $\mathcal{F}_{\mathsf{INIT}}$ and receives the stake distribution $\mathbb{S}_0$ as well as the random nonce η via the message $(\mathsf{genblock}, sid, \mathbb{S}_0, \eta)$, then sets $\mathcal{C}_i = B_0 = (\mathbb{S}_0, \eta)$ and its initial state $\mathsf{state}_i = H(B_0)$
2. **Chain Extension.** The stakeholder U_i with a relative stake α_i proceeds as follows. For each slot $\mathtt{sl}_j$, upon receiving data d from the environment $\mathcal{Z}$, proceed as follows:
   (a) Let $\mathbb{C}$ be the set of all chains collected from network, then
      i. Prune blocks belonging to future slots and verify that for every chain $\mathcal{C}' \in \mathbb{C}$, and every block $B_\ell = (\mathsf{state}_\ell, d_\ell, \mathtt{sl}_\ell, B_{\pi,\ell}, \sigma_{j,\ell}) \in \mathcal{C}'$ with its parent block $B_{\ell-1} = (\mathsf{state}_{\ell-1}, d_{\ell-1}, \mathtt{sl}_{\ell-1}, B_{\pi,\ell-1}, \sigma_{j,\ell-1}) \in \mathcal{C}'$, it holds that the stakeholder who created it is in the slot leader set of slot $\mathtt{sl}_\ell$ as follows
         A. Parse $B_{\pi,\ell}$ as (U_s, y, π) for some s, and compute $\delta_\ell = \mathtt{sl}_\ell - \mathtt{sl}_{\ell-1}$
         B. Check that U_s is the corresponding leader of the slot $\mathtt{sl}_\ell$ by sending $(\mathsf{Verify}, sid, \eta || \mathtt{sl}_\ell, y, \pi, v_s^{\mathrm{vrf}})$ to $\mathcal{F}_{\mathsf{VRF}}$ and receiving $(\mathsf{Verified}, sid, \eta || \mathtt{sl}_\ell, y, \pi, 1)$ and $y < 2^{\ell_{\mathrm{VRF}}} \phi(\delta_\ell, \alpha_s)$
         C. Check that $\sigma_{j,\ell}$ is a valid signature from U_s by sending $(\mathsf{Verify}, sid, (\mathsf{state}_\ell, d_\ell, \mathtt{sl}_\ell, B_{\pi,\ell}), \mathtt{sl}_\ell, v_s^{\mathrm{kes}}, \sigma_{j,\ell})$ to $\mathcal{F}_{\mathsf{KES}}$ and receiving $(\mathsf{Verified}, sid, (\mathsf{state}_\ell, d_\ell, \mathtt{sl}_\ell, B_{\pi,\ell}), \mathtt{sl}_\ell, 1)$
      ii. Run subroutine $\mathsf{maxvalid}$ over the set of chains from network $\mathbb{C}$ and his local chain $\mathcal{C}_i$, i.e., $\mathcal{C}' = \mathsf{maxvalid}(\mathbb{C} \cup \mathcal{C}_i)$, set $\mathcal{C}_i := \mathcal{C}'$ and $\mathsf{state}_i = H(\mathrm{head}(\mathcal{C}_i))$
   (b) The stakeholder U_i sends $(\mathsf{EvalProve}, sid, \eta || \mathtt{sl}_j)$ to $\mathcal{F}_{\mathsf{VRF}}$, receiving $(\mathsf{Evaluated}, sid, y, \pi)$. Let $\mathtt{sl}$ be the slot where the $\mathsf{head}(\mathcal{C}_i)$ was mined, the stakeholder U_i then computes $\delta = \mathtt{sl}_j - \mathtt{sl}$, and checks if $y < 2^{\ell_{\mathrm{VRF}}} \phi(\delta, \alpha_i)$.
   (c) If yes, then generates a new block $B = (\mathsf{state}_i, d, \mathtt{sl}_j, B_\pi, \sigma)$ where state_i is the current state of U_i, $d \in \{0,1\}^*$, $B_\pi = (U_i, y, \pi)$ and σ is a signature of $(\mathsf{state}_i, d, \mathtt{sl}_j, B_\pi)$ at slot $\mathtt{sl}_j$ from $\mathcal{F}_{\mathsf{KES}}$. Compute $\mathcal{C}_i = \mathcal{C}_i || B$ and $\mathsf{state}_i = H(\mathsf{head}(\mathcal{C}_i))$, then diffuse $\mathcal{C}'$
3. **Signing Transactions.** On message $(\mathsf{sign\text{-}tx}, sid', tx)$ from the environment, U_i sends $(\mathsf{Sign}, sid', U_i, tx)$ to $\mathcal{F}_{\mathsf{DSIG}}$ and then receives $(\mathsf{Signature}, sid', tx, \sigma)$. Then U_i sends $(\mathsf{signed\text{-}tx}, sid', tx, \sigma)$ to the environment.

Fig. 6: The Taktikos protocol. Highlights emphasize changes in construction from Ouroboros Praos [13].