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

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

[1] Topl, Austin, USA
{a.schutza,h.behrens}@topl.co
[2] Rice University, Houston, USA
james.aman@rice.edu
[3] FYEO, Inc., Jacksonville, USA
t.duong@gofyeo.com

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 by favoring those who have committed more stake. This eligibility threshold controls which parties may propose a block in a given slot, with easier thresholds increasing 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, finding that the dynamic regime regularizes slot intervals and improves block throughput. The pseudo-predictable nature of the protocol also penalizes covert attacks, simultaneously increasing security. We compare Ouroboros Taktikos to Ouroboros Praos and show that the addition of local dynamic difficulty improves throughput by ∼2.9× and reduces 99$^{\text{th}}$ percentile block latency by ∼5.7× 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

J. Chen et al. (Eds.): BlockSys 2023, CCIS 1897, pp. 269–283, 2024.
https://doi.org/10.1007/978-981-99-8104-5_20

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 [21], uses computational resources to secure proposer selection. Given the high power consumption of the 'work', some authors have expressed concerns [18,23,24] over the sustainability of PoW. Alternatives such as proof-of-stake (PoS) [2,5,7,9,10,12,15,17] 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 [19]). To address these cases, chain selection rules such as [10] limit the impact of forks, while other supplemental techniques improve bootstrapping [2] or clock synchronization [3].

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* [2,4,10,17]. Participants agree on time using existing approaches such as NTP, or via more decentralized schemes as in [3]. 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 [13] to more closely match their consistency bound.

In the Ouroboros family of protocols [17], 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*[1], solves these problems and improves upon

[1] From the Greek τακτικός, ordering or arranging, especially in a tactical sense.

Ouroboros Praos [10] 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. We propose one such function, the *snowplow curve*, that offers useful properties aligning with typical blockchain desiderata. 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.

Security Against Grinding Attacks. The conditional probabilities introduced by dynamic slot eligibility provide an additional degree of freedom to adversarial participants. While an adversary may discover a valid eligibility, they may choose to keep it private and reveal it later, when its relative position to other proposed blocks may trigger a reorganization. To address these concerns, we approach the challenge from two directions. First, we present a novel chain selection protocol compatible with Taktikos that balances these considerations. Second, we evaluate adversarial behavior in both the bounded and unbounded cases to determine if this new freedom produces any advantage. Finally, we analyze and bound the adversarial power stemming from these changes, contextualizing the advantage relative to our consistency bound.

Implementation and Empirical Comparison. We further implement our protocol and empirically evaluate its performance against existing approaches. Specifically, we compare with Praos [10], 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$2.9$\times$ increase in block throughput, and a $\sim$5.7$\times$ decrease in 99^{th} percentile block latency.

2 Related Work

Ouroboros [17] forms the foundation of Nakamoto-style probabilistic proof of stake and presents a formal framework for reasoning about blockchain consistency. Subsequent variants [2,3,10] 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. We build on this foundation to introduce the local dynamic difficulty mechanism, which to the best of our knowledge does not appear in any existing PoS scheme. Similarly, our assessment of grinding extends previous work [6,16], 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 [20]. 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. Other temporal approaches, such as proof-of-elapsed-time, share some similarities with our approach but rely on trusted execution modules such as Intel's SGX. Later work [8] has found that such hardware-based approaches contain vulnerabilities; our proposed approach does not require any secure hardware.

Evaluating the adversarial advantage in proof of stake systems, in particular from the perspective of the grinding adversary [5], has thus far implicitly assumed a static eligibility threshold. Markovian approaches such as those used in [17] or [14] therefore do not directly apply to the conditional probabilities that emerge from LDD. Simulation-based approaches help to bridge this gap, while leaving the door open to future work providing a stronger theoretical security analysis.

3 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 [10], a potential minter generates a test nonce $y_p \in {}^{\{0,1\}^{\ell_{\mathrm{VRF}}}}\!/_{2^{\ell_{\mathrm{VRF}}}}$, 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 [2].

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 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 for a slot interval δ with relative stake α:

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

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

$$1 - \phi\left(\delta_\ell, \sum_P \alpha_p\right) = \prod_P \big(1 - \phi(\delta_\ell, \alpha_p)\big) \tag{2}$$

Table 1. Key Notation

Δ	Network Delay	δ	Slot Interval
α	Relative Stake	p	Staking Party
F_A	Minimum Difficulty	F_B	Baseline Difficulty
ψ	Slot Gap	γ	Recovery Threshold

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 $\mathtt{sl}_{\ell-1}$, a set of forging parties P, a specific party $p \in P$, and its relative stake α_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 $f(\delta)$ returns 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 (Fig. 1).

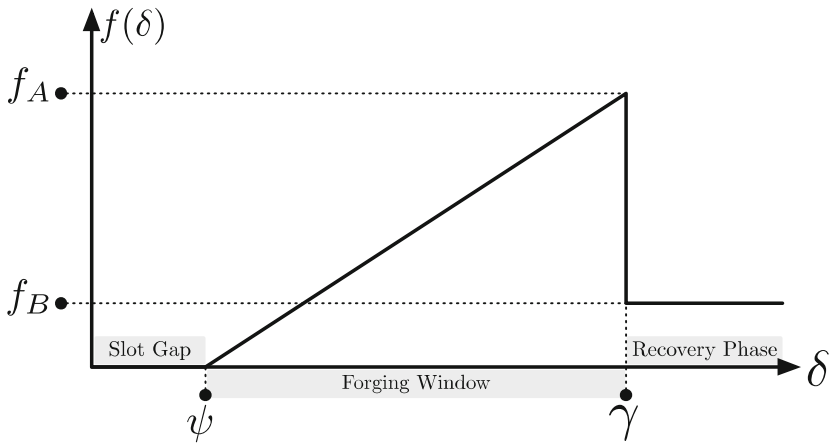


Fig. 1. Snowplow curve parameterization, visualized

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. Refer to Fig. 1 for a visual representation of Eq. 3. 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 [10]. A simplified form of $f(\delta)$ setting $\psi = 0$ and $\delta < \gamma < 1 + \frac{1}{c}$ is defined as $f(\delta) = c\delta$ where $c = f_A/\gamma$. 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 $0 < \delta < \frac{1}{c} + 1$, and by choosing a $\gamma < \frac{1}{c}$ an arbitrarily small portion of blocks fall in the recovery phase on the honest-majority tine. The proportion of filled slots corresponds to the expectation value of $f(\delta)$, which we call f_{eff}.

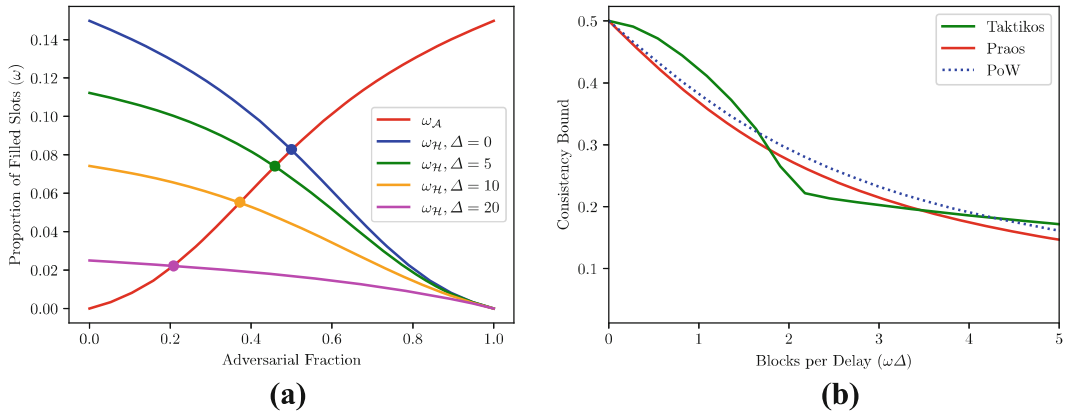
Intuitively, this curve rewards well-synchronized nodes with increased block production without requiring additional messaging, while still allowing out-of-date participants to catch up if needed. We now use the snowplow curve to define a revised consensus protocol that leverages LDD for its leader election.

The Consensus Protocol. Using LDD, we extend Ouroboros Praos [10] to a new protocol, *Ouroboros Taktikos*. Taktikos operates in the hybrid model with the functionalities $\mathcal{F}_{\mathsf{INIT}}$, $\mathcal{F}_{\mathsf{VRF}}$, $\mathcal{F}_{\mathsf{KES}}$, $\mathcal{F}_{\mathsf{DSIG}}$. 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 Fig. 5, which highlights key differences versus Praos.

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 [2,10]. 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 modification transforms Δ-divergences to unique convergence opportunities. The algorithm `maxvalid-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.

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

```
1  C ← C_loc;
2  for i ∈ {1, ..., j} do
3      if IsValidChain(C_i) then
4          if C_i forks from C at most k blocks then
5              if |C_i| = |C| then
6                  sl_i ← maxvalid-slot(C_i);
7                  sl_l ← maxvalid-slot(C);
8                  if sl_i < sl_l then
9                      C ← C_i;
10                 if |C_i| > |C| then
11                     C ← C_i;
12 return C
```



(a) (b)

Fig. 2. (a) Effective honest throughput with varying network delay. (b) Consistency bound comparison in terms of blocks per delay interval, with PoW trend from [13].

4 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 [6,16]. 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 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.

Consistency. Figure 2(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. 2(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 Fig. 2(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.

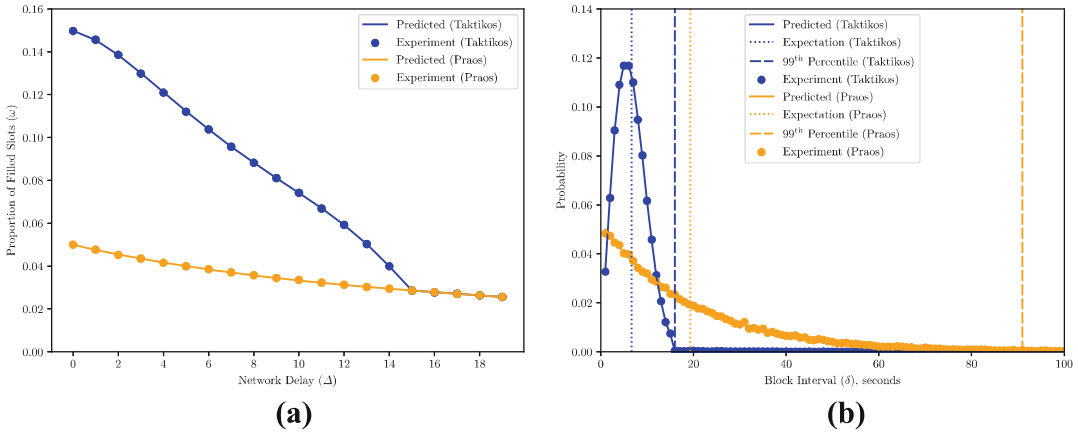


Fig. 3. Experimental comparison between Taktikos and Praos staking procedure. (a) Maximum expected chain growth as network latency increases; higher ω is better. (b) Probability density distribution of block intervals; lower δ is better.

Examining Fig. 2(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 [13] serves as our benchmark for comparison to the literature. Our analytic model for Praos aligns with the behavior from [10], 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.

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. 3, for the chosen parameterization of Taktikos[2] 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 Fig. 3(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 Fig. 3(a) when $\Delta \geq 15$. Over a range of realistic network latencies, $\Delta \sim 0 - 2$ seconds [22], 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 [11].

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 Fig. 3(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, typically seconds. 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$2.9$\times$ ($\frac{19.3}{6.67}$) more quickly and $\sim$5.7$\times$ ($\frac{91}{17}$) more predictably under Taktikos due to the shorter-tail distribution.

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.

[2] $f(\delta) \rightarrow (\psi = 0, \gamma = 15, f_A = 0.5, f_B = 0.05)$.

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.

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

Require : $y_i \in [0, 1]$ for $y_i \in \mathcal{Y}$, $r \in [0, 1]$, $f : \mathbb{N} \rightarrow [0, 1]$
$L \leftarrow |\mathcal{Y}|$; $\ell_{\max} \leftarrow 0$; $B \leftarrow [(0, 0)]$;
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 $\ell_{\max}/L$

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 Eq. 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 $f : \mathbb{N} \rightarrow [0, 1]$. The grinding frequency is given by taking the limit as $L \rightarrow \infty$.

The grinding frequency is given by the limit of Algorithm 2 as $L \rightarrow \infty$ such that

$$\omega_g = \lim_{L \rightarrow \infty} \mathsf{GrindingSim1}(\mathcal{Y}, r, f) \tag{4}$$

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 is an emergent property, varying with adversarial stake and the difficulty curve.

Algorithm 3: 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 $\ell_{\max}/L$;

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

$$\mathsf{GrindingSim1}(\mathcal{Y}, r, f) = \lim_{d \to \infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f) \tag{5}$$

and as a first-order approximation

$$\omega_g \approx \lim_{L \to \infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f)\big|_{d>1} \tag{6}$$

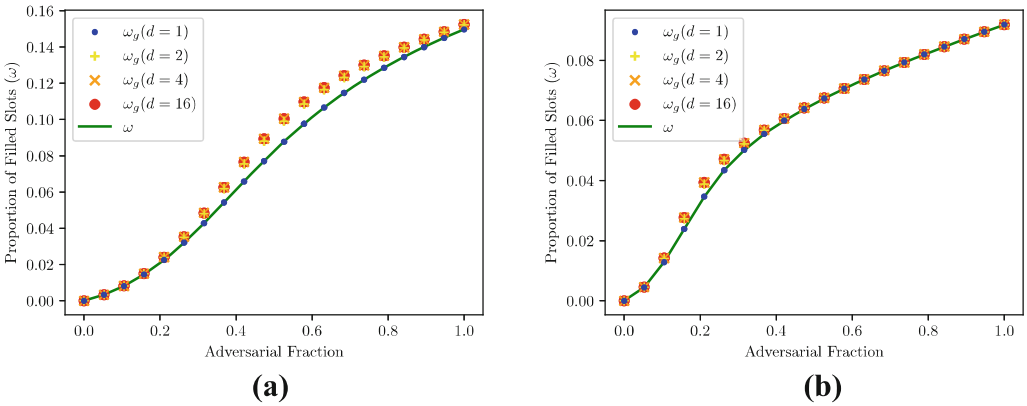


Fig. 4. Grinding advantage by adversarial fraction for (a) $\gamma = 15$. (b) $\gamma = 40$.

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

$$\omega = C\left[\sum_{\delta=0}^{\infty} \delta\pi_\delta\left(1-\left(1-f(\delta)\right)^r\right)\right]^{-1} \tag{7}$$

where π_δ is the stationary distribution and C is a normalization constant:

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

Figure 4 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 $d = 1$:

$$\omega \approx \lim_{L\to\infty} \mathsf{GrindingSim2}(\mathcal{Y}, r, d, f)\big|_{d=1} \tag{8}$$

The honest production rate ω, is predicted with Eq. 7. While the adversarial grinding curves ω_g, are given by evaluating Eq. 6 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. Note that, although the total AUC of $\gamma = 15$ exceeds that of $\gamma = 40$, most falls at a higher adversarial fraction beyond the consistency bound and is therefore not relevant.

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 added benefit in going from $d = 4$ to $d = 16$ proves negligible despite a much greater cost to the adversary. Therefore, while Taktikos does provide a small increase in adversarial power, the size of that advantage remains tightly bounded while other performance gains more than compensate for this tradeoff.

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 Fig. 2(a), we present this bound in terms of block proposals per delay interval, which varies based on block frequency and network delay.

To illustrate the challenge of direct comparison, we will describe a motivating scenario. Assume for the sake of example that all protocols assume a network delay of five seconds. Similarly, we will take the mean block frequency values from each protocol as their respective canonical frequencies: 6.67 s for Taktikos, and 19.29 s 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 Fig. 2(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

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. 5. The Taktikos protocol. Highlights emphasize changes in construction from Ouroboros Praos [10].

Praos' consistency bound while substantially improving performance (as detailed in Sect. 4).

Note that the chosen network delay and block frequency values were selected for convenience of comparison. In practice, these values will vary dynamically based on network conditions and stochastic variability, so fixed-point comparisons provide only a small window into protocol behavior. As network delay falls, these points approach 0 and differences in consistency bound decrease while throughput improvements persist.

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 random chance and 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 unused eligibility. Thus, even when tracking a large number of deep tines, nearly all must be discarded and the small fraction remaining provide the adversary only a modest grinding advantage.

5 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 propose a novel chain selection protocol and assess the effect that grinding adversaries have on protocol security in both the bounded and unbounded cases. We implement this protocol and empirically evaluate its performance against state-of-the-art, finding a $\sim$2.9$\times$ improvement in block throughput and a $\sim$5.7$\times$ reduction in 99^{th} percentile block latency. 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. The cardano node (2022). https://github.com/input-output-hk/cardano-node
2. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability. In: CCS (2018)
3. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros chronos: permissionless clock synchronization via proof-of-stake. Technical report (2019)
4. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Dynamic ad hoc clock synchronization. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 399–428. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_14
5. Bagaria, V., et al.: Proof-of-stake longest chain protocols: security vs predictability (2020). https://doi.org/10.48550/arXiv.1910.02218

6. 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)
7. Buterin, V., Griffith, V.: Casper the friendly finality gadget. arXiv:1710.09437 [cs] (2019)
8. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: On security analysis of proof-of-elapsed-time (PoET). In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 282–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_19
9. 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.) FC 2019. LNCS, vol. 11598, pp. 23–41. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32101-7_2
10. 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.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 66–98. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_3
11. Dembo, A., et al.: Everything is a race and Nakamoto always wins. In: CCS (2020)
12. Fan, L., Zhou, H.S.: A scalable proof-of-stake blockchain in the open setting. Cryptology ePrint Archive (2017)
13. Gaži, P., Kiayias, A., Russell, A.: Tight consistency bounds for bitcoin. In: CCS (2020)
14. Gaži, P., Ren, L., Russell, A.: Practical settlement bounds for longest-chain consensus. Cryptology ePrint Archive (2022). https://eprint.iacr.org/2022/1571
15. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: SOSP (2017)
16. Kiayias, A., Quader, S., Russell, A.: Consistency of proof-of-stake blockchains with concurrent honest slot leaders. In: ICDCS (2020)
17. 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. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
18. Küfeoğlu, S., Özkuran, M.: Bitcoin mining: a global review of energy and power demand. Energy Res. Soc. Sci. **58**, 101273 (2019)
19. 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.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 297–315. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_17
20. Lorenz, J., Lorenz, D.A.: On conditions for convergence to consensus. IEEE Trans. Autom. Control **55**, 1651–1656 (2010)
21. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Technical report (2008)
22. Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In: IEEE UIC (2016)
23. O'Dwyer, K.J., Malone, D.: Bitcoin mining and its energy footprint (2014)
24. 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.) RAID 2018. LNCS, vol. 11050, pp. 184–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00470-5_9