Practical Settlement Bounds for Proof-of-Work Blockchains

Peter Gaži
IOHK Research
peter.gazi@iohk.io

Ling Ren
University of Illinois,
Urbana-Champaign
renling@illinois.edu

Alexander Russell
University of Connecticut
IOHK Research
acr@uconn.edu

ABSTRACT

Nakamoto proof-of-work ledger consensus currently underlies the majority of deployed cryptocurrencies and smart-contract blockchains, especially when measured in carried value. While a long and fruitful line of work studying the provable security guarantees of this mechanism has succeeded to identify its exact security region—that is, the set of parametrizations under which it possesses *asymptotic* security—the existing theory does not provide concrete settlement time guarantees that are tight enough to inform practice.

In this work we provide a new approach for obtaining such settlement-time guarantees that provides strong, concrete bounds suitable for reasoning about deployed systems. This rigorous framework yields an efficient computational method for computing explicit bounds on settlement time as a function of the primary parameters: honest and adversarial computational power and a bound on network delays. Our framework simultaneously provides upper and lower bounds, which permits an immediate means for evaluating the strength of the results. We implement this computational method and provide a comprehensive sample of concrete bounds for several settings of interest.

For Bitcoin, for example, our explicit upper and lower bounds are within 90 seconds of each other after 1 hour of settlement delay with 10 second networking delays and a 10% adversary. In comparison, the best prior result has a gap of 2 hours in the upper and lower bounds with the same parameters.

1 INTRODUCTION

Nakamoto proof-of-work consensus, introduced in the 2008 Bitcoin white paper [17], is the basic algorithmic framework supporting the sensational Bitcoin and Ethereum blockchains. This charmingly simple protocol has inspired a large body of analytic work which—after over a decade of attention—has finally settled the *security region* of the protocol: specifically, two independent recent articles [5, 9] determine the exact conditions on honest hashing power, adversarial hashing power, and network delays under which the protocol achieves consensus. Pleasingly, their conclusion rigorously reaffirms the original intuition of the Bitcoin white paper.

In greater detail, these works determine the "asymptotic security region" of the protocol: they exactly identify the region of critical parameters (honest and adversarial hashing power, network delays) under which the probability of a consistency failure has the form $\exp(-\Omega(t))$, where t is the amount of time a given transaction has persisted in the blockchain and the asymptotic notation hides constants that depend on the critical parameters. Despite the interesting theoretical insights offered by the determination of this region and the analytic techniques that justify it, such asymptotic guarantees tell us very little about concrete settlement times because they don't provide control of the constants in the $\Omega(t)$ expression above (or provide any alternative means for establishing explicit settlement bounds). Indeed, their proof techniques are intentionally optimized for simplicity over precision. Even so, both papers present intricate and lengthy probabilistic analyses. This state of affairs is especially frustrating as it leaves conspicuously unanswered the most fundamental question faced by users of deployed blockchains:

How long must I wait for a transaction to settle?

One prominent feature of Nakamoto consensus is that the settlement question has a parametric answer: a block (and its transactions) achieves higher certainty with longer waiting times. The ideal answer would thus determine the exact probability of a settlement failure as a function of elapsed time or other observable phenomenon such as the number of subsequent blocks amassed on top of the block of interest.

Despite great practical significance, the above concrete question has not been adequately answered. The Bitcoin white paper [17] analyzed the transaction settlement time under a specific attack called the *private mining* attack. This approach of analyzing specific attacks was adopted by several follow-up works [10, 21]. However, it is clearly desirable to provide a more comprehensive answer: the gold standard for any protocol security analysis is to consider a well-defined and widely accepted model that puts limitations on the adversary without prescribing its concrete actions, and then prove that the protocol remains secure against *any* adversary covered by that model.

More concretely, the widely adopted model used for analyzing Nakamoto consensus that we also employ in this work, gives the adversary the ability to adaptively delay any messages sent by honest players and centrally control a collection of corrupt parties that may deviate arbitrarily from the protocol (by, for example, withholding blocks they create, creating private chains, etc.). To make the adversary more powerful, corrupt parties are assumed to be connected by a zero-latency network so that they can act with perfect knowledge of each other's states—thus the adversary can be characterized by a single entity with the amassed hashing power of all corrupt parties. On the other hand, honest players are assumed to follow the protocol to the letter (see Section 2.1), and, as standard, we assume an upper bound on the fraction of hashing power controlled by corrupted parties. In particular, our model does not cover attacks exploiting rational behavior of parties, such as the selfish-mining attacks [7], beyond the standard trick of considering such parties adversarial.

The main sticking point in analyzing the above question in such general models is accounting for possible network delays. Indeed, if one is content to assume an instantaneous or lockstep-synchronous network, two recent works [1, 5] do provide an exact analysis of consistency failures for proof-of-work blockchains. However, a practically relevant analysis needs to reflect network delays. As

mentioned above, most existing works that analyze blockchains in a general model with network delays ([3, 5, 8, 9, 12, 14, 18, 19, 22]) only give asymptotic bounds and do not directly speak to the question. Some recent works derive concrete security bounds by working out the constants offered by these asymptotic analyses ([3, 14]), but the resulting settlement bounds are very weak; in particular, for Bitcoin the results come to thousands of hours—orders of magnitude larger than what is used in practice. Only recently, Li et al. [15] derived the first practically viable settlement time upper bounds; for Bitcoin, for example, these results are a few hours larger than corresponding lower bounds.

1.1 Our results

We lay out a new proof technique for analyzing consistency of proof-of-work blockchains with an eye toward explicit settlement estimates. Our techniques result in a polynomial-time algorithm for computing explicit consistency guarantees. Our methods simultaneously yield upper and lower bounds for the probability of settlement failure, and thus provide an immediate means for direct evaluation. For example, when applied to Bitcoin with 10 second networking delays and a 10% adversary, our technique provides bounds that are within 90 seconds of optimality for a 1-hour waiting time. In greater detail, our contribution is twofold:

(1) **A rigorous framework for practical proof-of-work settlement bounds.** We devise a function **B** with a "dynamic programming" flavor that, given a description of a sequence of honest and adversarial mining successes throughout a fixed execution of the protocol (or a portion around the inclusion of some transaction tx of interest), addresses the question whether this given sequence of mining successes would permit, under arbitrary adversarial behavior and bounded network delays, an execution outcome that would lead to a settlement violation for tx. The form of the function **B** is a consequence of a broader theory that we develop along the way (more on that later), which allows us to rigorously prove that the output of **B** will never err in one direction: whenever it concludes that the settlement violation is impossible, this conclusion is correct.
(2) **Concrete failure probabilities for deployed systems.** The structural details of the quantity **B**—which yields a recursive expansion—permits it to be easily evaluated on a random variable (reflecting a schedule of mining successes), rather than a fixed schedule. We then consider the standard random schedule where honest and adversarial successes are independent Poisson processes parametrized by the respective mining powers. The resulting random variable computed by **B** then immediately yields an upper bound on the probability of a settlement violation in an execution with the respective distribution of mining power and the assumed upper bound on network delays. The "memoryless" property of the Poisson distribution permits us to compute this distribution exactly and efficiently.

This framework permits us to offer striking improvements over the best previous work—typically by a factor of 10 or more—in explicit settlement times for both the Bitcoin setting (with long interblock arrival times) and the Ethereum setting (with short interblock arrival times). In both settings, the final conclusions are within minutes of optimality. See Section 1.3 below for more detailed discussion.

1.2 A survey of the analysis

We capture the schedule of mining successes and the output of a concrete execution by a *characteristic string* and a *fork* respectively, two notions introduced for this purpose in the context of proof-of-stake [11] and adapted to PoW in [1, 9]. Our analysis departs almost immediately from [9] by shifting its focus to *serialization*. Given a sequence of block creation events, the longest-chain rule algorithm will produce a "hash tree" of blocks, where edges are given by the predecessor hash links in each generated block. The partial order assigned to blocks by this tree may naturally be inconsistent with the "real" linear order in which they were generated due both to network delays and decisions by adversarial players to postpone delivery of their blocks. Of course, there is always a reordering of the block creation events for which the tree has a simple "temporally consistent" explanation—such a reordering is a *serialization*. Motivated by the fact that the theory is tractable in the lockstep-synchronous case, we study the serializations that can arise in the setting with network delays, and then rely on the lockstep results to analyze the serialized executions.

Ultimately, we are interested in studying the longest chain rule in a continuous-time model $C[\Delta_r]$ where honest parties and the adversary both create proofs of work according to independent Poisson processes with rates r_h and r_a, respectively, and the adversary may selectively delay honest block delivery by up to Δ_r time ("r" stands for "real"). One way to capture this setting is to consider discrete slots corresponding to short intervals of length t ($t \ll \Delta_r$) and to appropriately adjust the networking model so that honest parties are not guaranteed to see messages that were sent to them at most $\Delta \triangleq \Delta_r/t$ slots ago. Call this model $\mathcal{D}[\Delta, t]$, where the two parameters record the maximum delay *in slots* and the duration of the slot, respectively. Note that the block-creation events are still governed by the same continuous Poisson processes, the discrete slot structure is put in place merely to reason about message delays. This is the approach taken in [9]; see the detailed discussion there on how taking $t \to 0$ makes this model approach $C[\Delta_r]$. However, one difficulty in tackling this model is in keeping track of the complex delay patterns that can occur when each message can be individually delayed by any number of up to Δ_r/t slots.

An alternative approach that we propose in this work is to introduce two "adjacent" discrete models (parameterized by the same real networking delay Δ_r). The first model $\mathcal{D}[0, \Delta_r]$ is effectively a lockstep-synchronous model, dividing time into relatively long slots of length Δ_r. Honest players producing blocks in the same slot are not apprised of each other's blocks, though the network is assumed to deliver all created (honest) blocks at the end of each slot. Of course, the adversary always operates with full knowledge of all adversarial and honest blocks produced in any slot. The second model $\mathcal{D}[1, \Delta_r]$ is similar, and a slot still represents a Δ_r-long time interval, but an honest block may now be delivered at the end of the slot following the one in which it was created; this effectively permits that some messages are delayed by up to $2\Delta_r$.

It is easy to observe that $\mathcal{D}[\Delta, t] \triangleq \mathcal{D}[\lceil \Delta_r/t \rceil, t]$ is "sandwiched" between these two models—informally,

$$\mathcal{D}[0, \Delta_r] \preceq \mathcal{D}[\Delta, t] \preceq \mathcal{D}[1, \Delta_r]$$
$$\downarrow (t \to 0)$$
$$C[\Delta_r]$$

where any valid execution in a model on the left-hand side of $\preceq$ is also valid in the model on the right-hand side, as the restriction on delays gets more permissive as we move to the right. Therefore, upper-bounding the probability of a mining schedule permitting a settlement failure in a later model also upper-bounds it in an earlier one; in other words, the $\mathcal{D}[0, \Delta_r]$ model settles more quickly than the model of interest $\mathcal{D}[\Delta, t]$, while $\mathcal{D}[1, \Delta_r]$ settles more slowly.

We first analyze consistency in $\mathcal{D}[0, \Delta_r]$ in Section 3. While this model appears to be more complicated than those of existing work [9], the analysis follows essentially the same lockstep trajectory and can be given fairly succinctly. We then shift our attention to $\mathcal{D}[1, \Delta_r]$, where the core technical difficulty lies, in Section 4. Our approach here is to bound consistency failures in $\mathcal{D}[1, \Delta_r]$ by showing how to synchronize an execution in this model *to an execution in* $\mathcal{D}[0, \Delta_r]$; we can then rely on the bounds for the simpler model obtained above.

A word on notation: In the rest of the paper, we reserve the symbol Δ to denote the maximum message delay *in slots*. Hence, in Section 3 (resp. 4), which employs the model $\mathcal{D}[0, \Delta_r]$ (resp. $\mathcal{D}[1, \Delta_r]$), we consider $\Delta = 0$ (resp. $\Delta = 1$). However, recall that in both these models, a slot itself has duration Δ_r.

We remark that our analysis does not consider difficulty adjustments that are present in PoW protocols. This is well justified by the fact that block settlement time is much shorter than the difficulty adjustment period (hours vs. weeks).

Finally, in Appendix A we also determine the *security region* in which our theory can be used to prove the security of the protocol except with negligible error.

1.3 Some example results generated by the theory

As mentioned in the abstract, our results provide very sharp estimates for Bitcoin in the region of practical interest; see Figure 1 which illustrates some of the time-based settlement results. With a 10% adversary and a bound of 10 seconds on networking delay, we obtain a settlement error of no more than 4.489% after one hour which can be directly compared with a lower bound of 4.261%. Notably, these results are "only minutes" apart: 90 seconds before the one hour mark, the lower bound is 4.494%. So the upper bound is less than 90 seconds away from optimal. Alternatively, our results yield a 99.8% settlement guarantee on blocks that appear buried by 6 other blocks.

In the case of Ethereum,[1] recall that blocks are comparatively small and have 13 second interblock time. With 2 second networking delays and a 10% adversary, our methods bound settlement failure by 0.1097% after four minutes. Our lower bound yields 0.02518% for the same period. As expected, we observe a larger gap than in the "nearly lockstep-synchronous" Bitcoin case. However, the result is still less than one minute behind the optimum: specifically, at the five minute mark the upper bound has fallen to 0.02219%. These results improve the settlement failure estimates of previous work by well over an order of magnitude in the regime of interest. See Figure 2 for a representative example of our results for Ethereum and a comparison with [15]; a more comprehensive discussion of both time-based and block-based settlement appears in Section 5 and Appendix C.

[1]Note that while Ethereum considers *uncle blocks* for difficulty recalculation and rewards distribution, these blocks do not affect its chain-selection rule, hence Ethereum is fully covered by our analysis.

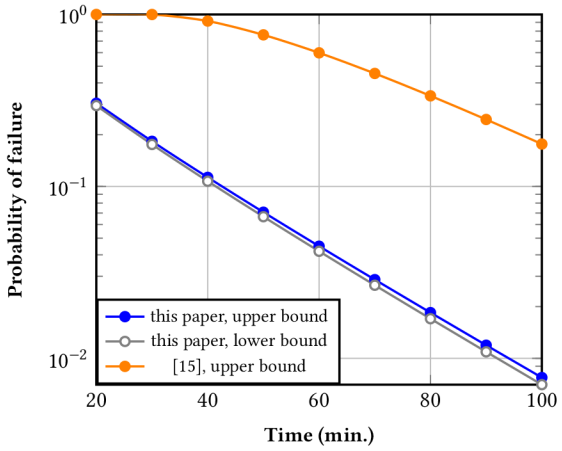

Figure 1: Our upper and lower bounds on settlement failure probability for Bitcoin with a 10% adversary and 10 second network delays; results from [15] included for comparison

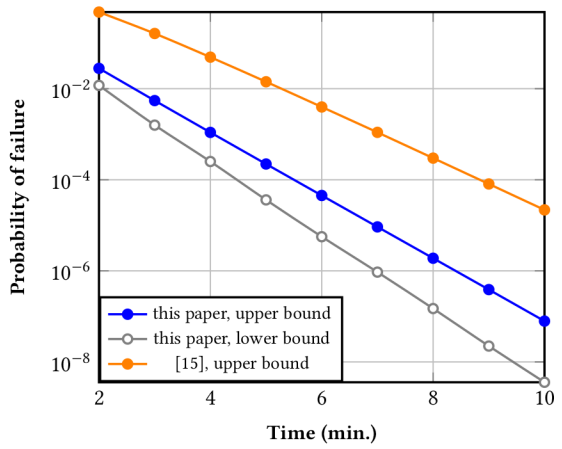

Figure 2: Our upper and lower bounds on settlement failure probability for Ethereum with a 10% adversary and 2 second network delays; results from [15] included for comparison.

2 PRELIMINARIES

Notation. Throughout the paper, $\mathbb{N} = \{0, 1, 2, \ldots\}$ denotes the set of natural numbers (including zero). For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$ (hence $[0] = \emptyset$). For a set X, we let $\mathcal{P}(X)$ denote the power set of X.

For a word of length n over alphabet Σ, we use the notation $w = w_1 \dots w_n \in \Sigma^n$. We denote by $w_{i:j}$ its subword $w_i w_{i+1} \dots w_j$, and $\#_a(w)$ denotes the number of occurrences of the symbol $a \in \Sigma$ in w. We denote by $\|$ the concatenation of words and by $\circ$ the concatenation of languages, i.e., $L_1 \circ L_2 \triangleq \{w_1 \| w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$.

2.1 Our Model of Proof-of-Work Blockchains

Our modeling of the protocol and its execution environment extends the model in [9], we summarize the model here for completeness.

A PoW blockchain protocol is carried out by a family of parties of two types: *honest* parties follow the protocol, *adversarial* parties that may diverge arbitrarily. All parties actively engage in searching for "proofs-of-work" (PoWs), which afford them the right to contribute to the ledger. For the purposes of analysis we treat time as divided into slots and use a *characteristic string* to indicate a summary of the outcome of the proof-of-work lottery in each slot, as well as the distribution of the lottery successes among honest and adversarial parties.

More concretely, our main alphabet of interest in this paper will be $\Sigma_\infty \triangleq \{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N}$. Intuitively, a single symbol $(s, a) \in \{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N}$ from this alphabet captures the outcome of the proof-of-work lottery in a given time slot, at a level of precision that will be most convenient for our treatment. Namely, the symbol $s \in \{0, \mathsf{h}, \mathsf{H}\}$ captures the number of honest successes as follows: 0 represents no honest successes, h represents one and only one honest success, and H denotes multiple honest successes in the considered slot. Finally, the natural number a simply captures the number of adversarial successes.

Note that a characteristic string symbol does *not* capture the full outcome of the lottery in a given slot: it merely describes the numbers of successes and their attribution to parties, but not their *ordering* within a slot. Looking ahead, it will be clear that our analysis implicitly assumes that this ordering is the best possible for the adversary, in line with our effort to obtain upper bounds on error probabilities.

For notational convenience when treating our imprecise accounting of honest successes described above, we define the following helper "rounding" function $\mathsf{round}_{\mathsf{H}} \colon \mathbb{N} \to \{0, \mathsf{h}, \mathsf{H}\}$. Let h be the number of honest successes in a given time slot and define

$$\mathsf{round}_{\mathsf{H}}(h) \triangleq \begin{cases} 0 & \text{if } h = 0\,, \\ \mathsf{h} & \text{if } h = 1\,, \\ \mathsf{H} & \text{if } h \geq 2\,. \end{cases} \tag{1}$$

We consider characteristic strings w (i.e., words) drawn from the set Σ_∞^L; these describe the outcomes of the proof-of-work lottery over a period of L slots. We write

$$w = (w_1, \dots, w_L) = ((s_1, a_1), \dots, (s_L, a_L))\,.$$

Let ε denote the empty characteristic string (i.e., $L = 0$).

The Bitcoin protocol calls for parties to exchange *blockchains*, each of which is an ordered sequence of blocks beginning with a distinguished "genesis block," known to all parties. Each proof-of-work success confers on that party the right to add exactly one block to an existing blockchain. (In fact, the party must identify the previous chain on which she wishes to build ahead of time, but this will not affect our analysis.) Honest parties follow the *longest-chain rule* which dictates that they always choose to add to the longest blockchain they have observed thus far and broadcast the result to all other parties. The basic dynamics of the system, with a particular characteristic string w and an adversary, can be described as follows.

Let $\mathcal{C}_t$ denote the collection of all blockchains created by time t and let $H(\mathcal{C}_t)$ denote the subset of all chains in $\mathcal{C}_t$ whose last block was created by an honest party. Set $\mathcal{C}_0 = \{G\}$, where G denotes the unique chain consisting solely of the genesis block. The genesis block is "honest"; thus $H(\mathcal{C}_0) = \mathcal{C}_0$. It is convenient to adopt the convention that $\mathcal{C}_{-t} = H(\mathcal{C}_{-t}) = \{G\}$ for any negative integer $-t < 0$. Then the protocol execution proceeds as follows. For each timestep $t = 1, 2, \dots$:

- Initiate $\mathcal{C}_t := \mathcal{C}_{t-1}$ and $H(\mathcal{C}_t) := H(\mathcal{C}_{t-1})$.
- If $w_t = (0, a)$, the adversary may repeat the following *adversarial iteration* a times: select a single blockchain C from $\mathcal{C}_t$ and add a block to create a new chain C', which is added to $\mathcal{C}_t$. $H(\mathcal{C}_t)$ remains unchanged.
- If $w_t = (\mathsf{h}, a)$, the same a adversarial iterations happen as above, but they are interleaved with a single *honest iteration* defined as follows: the adversary may select any collection of chains $\mathcal{V}$ for which $H(\mathcal{C}_{t-1-\Delta}) \subseteq \mathcal{V} \subseteq \mathcal{C}_t$. This is the "view" of the honest player, who applies the longest chain rule to $\mathcal{V}$, selects the longest chain $L \in \mathcal{V}$ where ties are broken by the adversary, and adds a new block to create a new chain L' that is added to $\mathcal{C}_t$ and also $H(\mathcal{C}_t)$.
- If $w_t = (\mathsf{H}, a)$, then the execution of a adversarial iterations is arbitrarily interleaved with at least two honest iterations.

In each time step t we also maintain the set of Δ-*dominant* chains $\mathcal{D}_t \subseteq \mathcal{C}_t$, determined entirely by $\mathcal{C}_t$ and $H(\mathcal{C}_{t-1-\Delta})$: namely, $\mathcal{D}_t$ is the set of all chains in $\mathcal{C}_t$ that are at least as long as the longest chain in $H(\mathcal{C}_{t-1-\Delta})$. The intuition behind the definition of Δ-dominant chains is that, in a time slot t, it is in principle possible for the adversary to manipulate an honest party into adopting any Δ-dominant chain, as the adversary is only obligated to deliver those chains in $H(\mathcal{C}_{t-1-\Delta})$ and the chains in $\mathcal{D}_t$ are at least as long as those in $H(\mathcal{C}_{t-1-\Delta})$.

Although we keep the presentation general, recall that as explained in the introduction, this work focuses on the two models $\mathcal{D}[0, \Delta_{\mathsf{r}}]$ and $\mathcal{D}[1, \Delta_{\mathsf{r}}]$, and hence always considers either $\Delta = 0$ or $\Delta = 1$.

This description implicitly places several constraints on the adversary; most notably, the only means of producing new chains is to append a block (associated with a proof-of-work success) to an existing chain. In practice, these constraints are guaranteed with cryptographic hash functions. Note that the synchrony assumption is reflected in the description of the honest iteration: the adversary is obligated to deliver all chains produced by honest players that are Δ slots old. Finally, we permit the adversary to have full view of the characteristic string during this process. Of course, in practice a Bitcoin adversary must make decisions "online," so our modeling only makes the adversary stronger.

While expressed as a game between the adversary and the honest players, considering that the adversary selects both the view $\mathcal{V}$ of each honest player and is empowered to break ties, the structure

of the resulting sequence of chains (that is, the directed acyclic graph naturally formed by the blocks) is determined entirely by the adversary and the characteristic string.

In the context of ledger protocols, one is usually interested in preserving two properties, *consistency* and *liveness*, formulated in [8, 13, 20]. Consistency means that once a block (or equivalently, a transaction within it) is *settled*, then it remains settled forever. We consider two settlement rules in this paper: a time-based one and a block-based one.

- **Consistency for time-based settlement; with parameter τ.** A block B that is mined before time ℓ and contained in some chain in $\mathcal{D}_t$ where $t \geq \ell + \tau$ is contained in every chain $C \in \mathcal{D}_{t'}$ for all $t' \geq t$.
- **Consistency for block-based settlement; with parameter k.** A block B that is mined before time ℓ and is k blocks deep in some chain in $\mathcal{D}_t$ is contained in every chain $C \in \mathcal{D}_{t'}$ for all $t' \geq t$.

Intuitively, the above settlement rules state that, when an honest player examines the longest chain to its knowledge at time t, it considers all blocks mined at least τ earlier (in the case of time-based settlement) or buried k blocks deep (in the case of block-based settlement) settled. Nakamoto-style blockchain protocols achieve probabilistic consistency. The focus of this paper is to bound the error probability (from both above and below) as a function of the settlement delay, i.e., of the parameters τ or k in the above two settlement rules, respectively.

We also remark that our definitions of consistency and its error probability above are applicable to individual blocks. One can also phrase the consistency as a global property of the protocol by requiring the above to hold for all blocks. However, the error probability of such a global consistency property will depend on the total running time of the blockchain protocol and is hard to characterize accurately.

For completeness, we also mention the liveness property [9].

- **Liveness; with parameter u.** For any two slots $t_1, t_2 > 0$ with $t_1 + u \leq t_2$, and any chain $C \in \mathcal{D}_{t_2}$, there is a time $t' \in \{t_1, \ldots, t_1 + u\}$ and a chain $C' \in H(C_{t'}) \setminus H(C_{t'-1})$ such that C' is a prefix of C.

2.2 Proof-of-Work Forks

We formally capture the above protocol dynamics by the combinatorial notion of a *fork*. It is a variant of the "fork" concept first considered for the proof-of-stake case in [2, 4, 11] and more recently also employed for PoW-analysis [1, 9].

Definition 2.1 (PoW Δ-fork). Let $\Delta, L \in \mathbb{N}$. A (PoW) Δ-*fork* for the string $w \in \Sigma_\infty^L$ is a directed, rooted tree $F = (V, E)$ with a pair of functions

$$\mathsf{l}_\# : V \to \{0, \ldots, L\} \qquad \text{and} \qquad \mathsf{l}_{\mathsf{type}} : V \to \{\mathsf{h}, \mathsf{a}\}$$

satisfying the axioms below. Edges are directed "away from" the root so that there is a unique directed path from the root to any vertex. The value $\mathsf{l}_\#(v)$ is referred to as the *label* of v. The value $\mathsf{l}_{\mathsf{type}}(v)$ is referred to as the *type* of the vertex: when $\mathsf{l}_{\mathsf{type}}(v) = \mathsf{h}$, we say that the vertex is *honest*; otherwise it is *adversarial*.

(A1) the root $r \in V$ is honest and is the only vertex with label $\mathsf{l}_\#(r) = 0$;

(A2) the sequence of labels $\mathsf{l}_\#()$ along any directed path is non-decreasing;

(A3) if $w_i = (s_i, a_i)$ then the number h_i of honest vertices of F with the label i satisfies $\mathsf{round}_{\mathsf{H}}(h_i) = s_i$, and there are no more than a_i adversarial vertices of F with the label i;

(A4) for any pair of honest vertices v, w for which $\mathsf{l}_\#(v) + \Delta < \mathsf{l}_\#(w)$, $\mathsf{len}(v) < \mathsf{len}(w)$, where $\mathsf{len}()$ denotes the depth of the vertex.

A Δ-fork abstracts a protocol execution with a simple but sufficiently descriptive discrete structure. Its vertices and edges stand for blocks and their connecting hash links (in reverse direction), respectively. The root represents the genesis block, and for each vertex v, $\mathsf{l}_\#(v)$ and $\mathsf{len}(v)$ denote the slot in which the corresponding block was created and the block's depth, respectively.

It is easy to see the correspondence between the above axioms and the constraints imposed in the protocol execution. In particular, (A1) corresponds to the trusted nature of the genesis block; (A2) reflects that the blocks' ordering in a chain must be consistent with the order of their creation; (A3) reflects that honest players produce exactly one block per PoW success, while the adversary might forgo a block-creation opportunity; finally (A4) reflects the fact that given sufficient time, as needed for block propagation in the network, an honest party will take into account the blocks produced by previous honest parties.

Definition 2.2 (Fork notation). We write $F \vdash_\Delta w$ to indicate that F is a Δ-fork for the string w. If $F' \vdash_\Delta w'$ for a prefix w' of w, we say that F' is a *subfork* of F, denoted $F' \sqsubseteq F$, if F contains F' as a consistently-labeled subgraph. A fork $F \vdash_\Delta w$ is *closed* if all leaves are honest. The trivial fork, consisting solely of a root vertex, is considered closed. The *closure* of a fork F, denoted $\overline{F}$, is the maximal closed subfork of F. We call two forks F_1 and F_2 *equivalent*, denoted $F_1 \equiv F_2$, if their underlying graphs and the $\mathsf{l}_{\mathsf{type}}(\cdot)$ functions are identical. Note that equivalent forks may only differ in their $\mathsf{l}_\#(\cdot)$ functions; whenever useful, we indicate the fork to which a labeling function belongs by a superscipt (e.g. $\mathsf{l}_\#^F(\cdot)$).

An individual blockchain constructed during the protocol execution is represented by the notion of a *tine*, defined next. Consequently, in later informal discussions we often identify a blockchain with its respective tine if no confusion can arise.

Definition 2.3 (Tines). A path in a fork F originating at the root is called a *tine* (note that tines do not necessarily terminate at a leaf). As there is a one-to-one correspondence between directed paths from the root and vertices of a fork, we routinely overload notation so that it applies to both tines and vertices. Specifically, we let $\mathsf{len}(T)$ denote the *length* of the tine, equal to the number of edges on the path; recall that $\mathsf{len}(v)$ also denotes the depth of a vertex. We sometimes emphasize the fork from which v is drawn by writing $\mathsf{len}_F(v)$. We further overload this notation by letting $\mathsf{len}(F)$ denote the length of the longest tine in a fork F. Likewise, we let $\mathsf{l}_\#(\cdot)$ apply to tines by defining $\mathsf{l}_\#(T) \triangleq \mathsf{l}_\#(v)$, where v is the terminal vertex on the tine T. We say that a tine is *honest* if the last vertex of the tine is honest. For a vertex v in a fork F, we denote by $F(v)$ the tine in F terminating in v.

For two tines T, T' of a fork F, we write $T \sim_\ell T'$ if the two tines share a vertex with a label greater or equal to ℓ.

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

Definition 2.4 (Fork trimming; dominance). For a string $w = w_1 \dots w_n$ and some $k \in \mathbb{N}$, we let $w_{\lceil k} = w_1 \dots w_{n-k}$ denote the string obtained by removing the last k symbols. For a fork $F \vdash_\Delta w_1 \dots w_n$ we let $F_{\lceil k} \vdash_\Delta w_{\lceil k}$ denote the fork obtained by retaining only those vertices labeled from the set $\{1, \dots, n-k\}$. We say that a tine T in F is Δ-*dominant* if $\mathsf{len}(T) \geq \mathsf{len}(\overline{F_{\lceil \Delta}})$ and simply call it *dominant* if Δ is clear from the context.

Observe that honest tines appearing in $F_{\lceil \Delta}$ are those that are necessarily visible to honest players at a round just beyond the last one described by the characteristic string. Correspondingly, the notion of a Δ-dominant tine corresponds to Δ-dominant chains as defined in Section 2.1.

2.3 Advantage and Margin

Definition 2.5 (Advantage α_F^Δ). For a Δ-fork $F \vdash_\Delta w$, we define the Δ-*advantage* of a tine $T \in F$ as

$$\alpha_F^\Delta(T) = \mathsf{len}(T) - \mathsf{len}(\overline{F_{\lceil \Delta}}) \,.$$

Observe that $\alpha_F^\Delta(T) \geq 0$ if and only if T is Δ-dominant in F.

Definition 2.6 (Margin β_ℓ^Δ). For $\ell \geq 1$, we define the Δ-*margin* of a fork F as

$$\beta_\ell^\Delta(F) = \max_{\substack{T^* \nsim_\ell T \\ T^* \text{ is } \Delta\text{-dominant}}} \alpha_F^\Delta(T) \,,$$

this maximum extended over all pairs of tines (T, T^*) where T^* is Δ-dominant and $T \nsim_\ell T^*$. We call the pair (T^*, T) the Δ-*witness tines for* F if the above conditions are satisfied; i.e., T^* is Δ-dominant, $T^* \nsim_\ell T$, and $\beta_\ell^\Delta(F) = \alpha_F^\Delta(T)$. Note that there might exist multiple such pairs in F, but under the condition $\ell \geq 1$ there will always exist at least one such pair, as the trivial tine T_0 containing only the root vertex satisfies $T_0 \nsim_\ell T$ for any T and $\ell \geq 1$, in particular $T_0 \nsim_\ell T_0$. For this reason, we will always consider β_ℓ^Δ only for $\ell \geq 1$.

We overload the notation and let

$$\beta_\ell^\Delta(w) = \max_{F \vdash_\Delta w} \beta_\ell^\Delta(F) \,.$$

We call a fork $F \vdash_\Delta w$ a Δ-*witness fork for* w if $\beta_\ell^\Delta(w) = \beta_\ell^\Delta(F)$; again many Δ-witness forks may exist for a string w.

Intuitively, $\alpha_F^\Delta(T)$ captures the length advantage (or deficit) of the tine T against the longest honest tine created at least Δ slots before the upcoming slot, which is hence now known to all honest parties. Consequently, $\beta_\ell^\Delta(F)$ records the maximal advantage of any tine T in F that potentially disagrees with some Δ-dominant tine T^* about the chain state up to slot ℓ. A negative $\beta_\ell^\Delta(F)$ hence indicates that the adversary cannot make an honest party holding T^* switch to any T that would potentially cause a revision of its ledger state up to slot ℓ. This connection between margin and consistency/settlement is exploited in previous work, for the PoW case it was made formal in [9, Lemma 1] in which the following fact is implicit:

Lemma 2.7 ([9]). *Consider an execution of a PoW blockchain for L slots as described in Section 2.1, resulting in a characteristic string $w = w_1 \dots w_L$. Let B be a block produced in slot $\ell \in [L]$, and let $t > \ell$ be such that B is contained in some chain $C \in \mathcal{D}_t$. If for every $t' \in \{t, \dots, L\}$ we have $\beta_\ell^\Delta(w_{1:t'}) < 0$ then B is contained in every $C' \in \mathcal{D}_{t'}$ for all $t' \in \{t, \dots, L\}$.*

This statement motivates our effort to upper-bound $\beta_\ell^\Delta(w)$ in the following sections.

Remark. One can define and study an analogous notion of consistency for protocols with unbounded lifetimes and, in fact, the explicit upper bounds we compute later in the paper reflect this stronger notion. Specifically, for a characteristic string $w = w_1 w_2 \dots$ and a finite ℓ, this requires that $\beta_\ell^\Delta(w_{1:t'}) < 0$ for all $t' \geq \ell$.

3 LOCKSTEP-SYNCHRONOUS ANALYSIS ($\Delta = 0$)

In this section we focus on the simpler, so-called lockstep-synchronous setting, where all messages are delivered at the end of the slot in which they were sent, this corresponds to $\Delta = 0$ and hence can be used to upper-bound errors in the model $\mathcal{D}[0, \Delta_r]$. Throughout the section, as no confusion can arise, we omit the index 0 and write $F \vdash w$, $\alpha_F()$, $\beta_\ell()$, in place of $F \vdash_0 w$, $\alpha_F^0()$, $\beta_\ell^0()$, respectively. Note that now $\alpha_F(T) = \mathsf{len}(T) - \mathsf{len}(\overline{F})$.

Our main goal will be to obtain a simple recursive description of the margin quantity $\beta_\ell(w)$ for a characteristic string $w \in \Sigma_\infty^*$. Looking ahead, we will obtain an exact characterization (Theorem 3.6) that will then serve us later in Section 4 when establishing bounds for the margin $\beta_\ell^1(w)$ in the case with delays $\Delta = 1$.

3.1 The Fully Serialized Setting ($\Sigma_{\mathsf{ser}} = \{\mathsf{h}, \mathsf{a}\}$)

We begin the analysis of the lockstep-synchronous setting by considering an additional simplification, assuming that the block creation is strictly serialized, i.e., exactly one block is created in each time slot. Specifically, we work with a reduced alphabet $\Sigma_{\mathsf{ser}} = \{(\mathsf{h}, 0), (0, 1)\}$ for characteristic strings, and use the abbreviations $\mathsf{h} = (\mathsf{h}, 0)$ and $\mathsf{a} = (0, 1)$; thus we treat characteristic strings over the alphabet $\{\mathsf{h}, \mathsf{a}\}$. The definition of fork remains unchanged.

The following exact characterization of β_ℓ in the lockstep (i.e., $\Delta = 0$), fully serialized (i.e., with alphabet Σ_{ser}) setting was given in prior work [1] and we present it here as an instructive starting point of our investigation. Recall that ε is the empty characteristic string.

Lemma 3.1 ([1, Lemma 1]). *Fix $\ell \geq 1$. We consider characteristic strings $w \in \Sigma_{\mathsf{ser}}^*$. By definition $\beta_\ell(\varepsilon) = 0$. In general, $\beta_\ell(w\mathsf{a}) = \beta_\ell(w) + 1$ and*

$$\beta_\ell(w\mathsf{h}) = \begin{cases} \beta_\ell(w), & \text{if } \beta_\ell(w) = 0 \text{ and } |w\mathsf{h}| < \ell, \\ \beta_\ell(w) - 1, & \text{otherwise.} \end{cases}$$

Thus, prior to round ℓ, β_ℓ performs a biased *barrier walk* with a barrier at 0; *after* round ℓ, it performs a standard biased random walk.

3.2 The Setting with Multi-honest Slots ($\Sigma_{\mathsf{mh}} = \{\mathsf{h}, \mathsf{H}, \mathsf{a}\}$)

We now slightly generalize the treatment of Section 3.1 and consider characteristic strings over an alphabet that allows for multiple honest successes in a single slot. Namely, we consider $\Sigma_{\mathsf{mh}} = \{(\mathsf{h}, 0), (\mathsf{H}, 0), (0, 1)\} \subset \Sigma_\infty$, and use the shorthands $\{\mathsf{h}, \mathsf{H}, \mathsf{a}\}$ for these three symbols, respectively. The definition of a fork again remains unchanged.

Lemma 3.2. *Fix $\ell \geq 1$. We consider characteristic strings $w \in \Sigma_{\mathsf{mh}}^*$. By definition $\beta_\ell(\varepsilon) = 0$. In general, $\beta_\ell(w\mathsf{a}) = \beta_\ell(w) + 1$ and*

$$\beta_\ell(w\mathsf{h}) = \begin{cases} \beta_\ell(w), & \text{if } \beta_\ell(w) = 0 \text{ and } |w\mathsf{h}| < \ell, \\ \beta_\ell(w) - 1, & \text{otherwise,} \end{cases}$$
$$\beta_\ell(w\mathsf{H}) = \begin{cases} \beta_\ell(w), & \text{if } \beta_\ell(w) = 0, \\ \beta_\ell(w) - 1, & \text{otherwise.} \end{cases} \tag{2}$$

Informally, the reason why H has a different effect on β_ℓ than h after slot ℓ is that if $\beta_\ell(w) = 0$, this means that there are two competing tines of the *same*, maximal length that can be served to an honest party; now the adversary can orchestrate things so that the two (or more) honest successes occurring in this slot contribute to both of these chains equally, and hence they don't improve the situation of the honest parties. We call this effect a *neutralization* of the honest success(es). Note that, in contrast, a unique honest success h improves the situation of the honest parties in the "tie" case of $\beta(w) = 0$, as it only extends one of the tines.

The proof of Lemma 3.2 is an extension of the proof of Lemma 3.1 that appeared in [1], accounting for presence of H symbols in the considered characteristic string, we provide it in full in Appendix B.

3.3 The General Case ($\Sigma_\infty = \{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N}$)

We finally consider the full alphabet $\Sigma_\infty = \{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N}$. Intuitively, our approach here is to assign to any "rich" characteristic string $w \in \Sigma_\infty^*$ a set of "possible serializations" $R_0(w) \subseteq \Sigma_{\mathsf{mh}}^*$ such that any fork over w can be interpreted (via relabeling) as a fork over one of these Σ_{mh}-serializations, and vice versa. This then allows to precisely characterize $\beta_\ell(w)$ in terms of $\beta_\ell()$ of these Σ_{mh}-serializations, which are already understood in Lemma 3.2.

Serialization of the general alphabet. We define a serialization mapping $R_0 \colon \Sigma_\infty \to \mathcal{P}(\Sigma_{\mathsf{mh}}^*)$ as follows:

$$\begin{aligned} R_0(0, k) &= \left\{\mathsf{a}^k\right\}, \\ R_0(\mathsf{h}, k) &= \left\{r \in \{\mathsf{a}, \mathsf{h}\}^* \mid \#_{\mathsf{h}}(r) = 1 \wedge \#_{\mathsf{a}}(r) = k\right\}, \\ R_0(\mathsf{H}, k) &= \left\{r \in \{\mathsf{a}, \mathsf{h}, \mathsf{H}\}^* \mid \#_{\mathsf{a}}(r) = k \wedge \right. \\ &\qquad \left. \wedge \left(\#_{\mathsf{h}}(r) \geq 2 \vee \#_{\mathsf{H}}(r) \geq 1\right)\right\}. \end{aligned}$$

Moreover, we naturally extend the mapping $R_0(\cdot)$ to strings $w = w_1 \dots w_n \in \Sigma_\infty^*$ by the convention

$$R_0(w) \triangleq R_0(w_1) \circ \cdots \circ R_0(w_n) \subseteq \Sigma_{\mathsf{mh}}^*.$$

Lemma 3.3. *Let $w \in \Sigma_\infty^n$ and $F \vdash w$. Then there is a characteristic string $w' \in R_0(w)$ and a fork $F' \vdash w'$ such that $F' \equiv F$.*

Proof. Consider the fragment of a fork $F \vdash w = w_1 \dots w_n \in \Sigma_\infty^n$ induced by vertices attributed to a particular symbol $w_i \in \Sigma_\infty$. This is a (potentially disconnected) forest of trees. Consider a topological sort of these trees, treated as a single directed acyclic graph. Then it is clear that the trees can be realized over the atomic characteristic string $w'_1 \dots w'_m \in R_0(w_i) \subseteq (\Sigma_{\mathsf{mh}})^m$, where m is the number of vertices appearing in the trees and $w'_i = \mathsf{l}_{\mathsf{type}}(v)$, where v is the vertex assigned the number i in the topological sort. The lemma follows. □

Lemma 3.4. *Let $w \in \Sigma_\infty^*$ and $w' \in R_0(w)$. Then for any fork $F' \vdash w'$ there exists a fork $F \vdash w$ such that $F \equiv F'$.*

Proof. Let v be a vertex in F' with $\mathsf{l}_\#(v) = j \in |w'|$, and let $i \in |w|$ be the index in $w = w_1 \dots w_{|w|}$ such that the j-th symbol in w' belongs to the expansion $R_0(w_i)$ of w_i. Then it suffices to set the label of v in F as $\mathsf{l}^F_\#(v) = i$. The correctness of this construction follows directly from the definition of R_0. □

Lemmas 3.3 and 3.4 immediately imply the following corollary.

Corollary 3.5. *Let $w \in \Sigma_\infty^*$. Then*

$$\beta_\ell(w) = \max_{w' \in R_0(w)} \beta_{\ell'}(w'),$$

where ℓ' is the appropriate index in w' corresponding to ℓ in w.

Proof. Let $F \vdash w$ be a witness fork, and let $w^* \in R_0(w)$ and $F^* \equiv F$ be such that $F^* \vdash w^*$ as guaranteed by Lemma 3.3. Let ℓ^* be the appropriate index in w^* corresponding to ℓ in w. We have

$$\beta_\ell(w) = \beta_\ell(F) = \beta_{\ell^*}(F^*) \leq \beta_{\ell^*}(w^*) \leq \max_{w' \in R_0(w)} \beta_{\ell'}(w')$$

where ℓ' is defined as in the statement of the lemma, establishing the first inequality.

For the opposite inequality, let

$$w^* \triangleq \arg \max_{w' \in R_0(w)} \beta_{\ell'}(w')$$

for ℓ' as defined in the statement, let ℓ^* be the respective value for w^*, and let $F^* \vdash w^*$ be its witness fork. Let $F \vdash w$ be the fork satisfying $F \equiv F^*$ as guaranteed by Lemma 3.4. Then

$$\max_{w' \in R_0(w)} \beta_{\ell'}(w') = \beta_{\ell^*}(w^*) = \beta_{\ell^*}(F^*) = \beta_\ell(F) \leq \beta_\ell(w)$$

as desired. □

Now we are ready to establish the main result of this section.

Theorem 3.6. *Fix $\ell \geq 1$. We consider characteristic strings $w \in \Sigma_\infty^* = (\{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N})^*$. By definition $\beta_\ell(\varepsilon) = 0$. In general,*

$$\beta_\ell(w(0, a)) = \beta_\ell(w) + a,$$
$$\beta_\ell(w(\mathsf{h}, a)) = \begin{cases} \beta_\ell(w) + a, & \text{if } \beta_\ell(w) = 0 \wedge |w| + 1 < \ell, \\ \beta_\ell(w) + a - 1, & \text{otherwise,} \end{cases}$$
$$\beta_\ell(w(\mathsf{H}, a)) = \begin{cases} \beta_\ell(w) + a, & \text{if } -a \leq \beta_\ell(w) \leq 0, \\ \beta_\ell(w) + a - 1, & \text{otherwise.} \end{cases}$$

Proof. The statements are shown independently for each case, always applying Corollary 3.5, the definition of the mapping R_0, and Lemma 3.2. Concretely, in the simplest case we have

$$\begin{aligned} \beta_\ell(w(0, a)) &= \max_{w' \in R_0(w(0,a))} \beta_{\ell'}(w') = \max_{w'' \in R_0(w)} \beta_{\ell'}(w''\mathsf{a}^k) \\ &= \max_{w'' \in R_0(w)} \beta_{\ell'}(w'') + k = \beta_\ell(w) + a. \end{aligned}$$

The other two cases are fully analogous, additionally taking into account subcases depending on the value of $\beta_\ell(w)$ and ℓ when invoking Lemma 3.2. □

4 ANALYSIS WITH DELAYS

We now move our attention to the case $\Delta = 1$. Contrary to the previous section, we will not derive an exact description of $\beta^1_\ell(w)$; nonetheless, we will define an easy-to-compute recurrent function $\mathbf{B}_\ell(w)$ that we show can give us a good upper-bound on $\beta^1_\ell(w)$.

4.1 Weak Serialization via Deferrals

We start by defining the set $\mathcal{D}_1(w)$ of so-called *deferrals* of w that will play a somewhat similar role in this section as the set of serializations $R_0(w)$ in Section 3. The important difference is that while R_0 partially serialized the block-creation events captured in w, deferrals have a different goal: they account for the possible 1-slot delay of these successes without actually fully serializing them. A deferral is hence still a characteristic string over the rich, unserialized alphabet Σ_∞.

Definition 4.1 (Realizations and deferrals). Consider a characteristic string $w = ((s_1, a_1), \ldots, (s_n, a_n)) \in \Sigma_\infty^n$. A *realization* of w is a string $r = ((h_1, a_1), \ldots, (h_n, a_n)) \in (\mathbb{N} \times \mathbb{N})^n$ where for each $i \in [n]$ we have $s_i = \mathsf{round}_{\mathsf{H}}(h_i)$. Let

$$r = ((h_1, a_1), \ldots, (h_n, a_n))$$
$$r' = ((h'_1, a'_1), \ldots, (h'_n, a'_n), (h'_{n+1}, a'_{n+1}))$$

be two realizations, where each (h_i, a_i) and (h'_i, a'_i) are elements of $\mathbb{N}^2$. We say that r' is a *1-deferral* of r if

(1) for each $t \in \{0, \ldots, n\}$, $\sum_{i=1}^{t} a_i \le \sum_{i=1}^{t+1} a'_i \le \sum_{i}^{t+1} a_i$, and
(2) for each $t \in \{0, \ldots, n\}$, $\sum_{i=1}^{t} h_i \le \sum_{i=1}^{t+1} h'_i \le \sum_{i}^{t+1} h_i$,

where we adopt the convention that $a_{n+1} = h_{n+1} = 0$. Finally, consider two characteristic strings

$$w = ((s_1, a_1), \ldots, (s_n, a_n)) \in \Sigma_\infty^n ,$$
$$w' = ((s'_1, a'_1), \ldots, (s'_n, a'_n), (s'_{n+1}, a'_{n+1})) \in \Sigma_\infty^{n+1} .$$

We say that w' is a 1-deferral of w if there are realizations r (of w) and r' (of w') so that r' is a 1-deferral of r. Let $\mathcal{D}_1(w)$ denote the set of all 1-deferrals of w. As we only consider 1-deferrals in this work, we sometimes simply call them deferrals.

The following lemma is an analogue of Lemma 3.3, showing that any 1-fork of w can be seen as a 0-fork of some 1-serialization of w.

Lemma 4.2. *Let $w \in \Sigma_\infty^n$ and $F \vdash_1 w$. Then there is a 1-deferral $w' \in \mathcal{D}_1(w)$ and an equivalent fork $F' \equiv F$ such that $F' \vdash_0 w'$.*

Proof. For any fork F we call a pair of vertices (u, v) a *violating pair in F* if $\mathsf{l}_{\mathsf{type}}(u) = \mathsf{l}_{\mathsf{type}}(v) = \mathsf{h}$, $\mathsf{l}^F_{\#}(u) < \mathsf{l}^F_{\#}(v)$ and $\mathsf{len}_F(u) \ge \mathsf{len}_F(v)$. Denote by $\mathcal{V}(F)$ the set of all violating pairs in F.

Now consider $w = ((s_1, a_1), \ldots, (s_n, a_n)) \in \Sigma_\infty^n$, with $s_i \in \{0, \mathsf{H}, \mathsf{h}\}$ and $a_i \in \mathbb{N}$ for each $i \in [n]$, and a fork $F \vdash_1 w$ as in the statement of the lemma. We first construct the string $w' \in \Sigma_\infty^{n+1}$ and show that $w' \in \mathcal{D}_1(w)$. Let V_i denote the set of vertices in F with $\mathsf{l}_{\#}$-label i, then for each $i \in [n+1] \cup \{0\}$, define

$$\widehat{\mathcal{H}}_i \triangleq \left\{u \in V_i \mid \mathsf{l}_{\mathsf{type}}(u) = \mathsf{h} \wedge [\exists v \in F\colon (u, v) \in \mathcal{V}(F)]\right\} ,$$
$$\mathcal{H}_i \triangleq \left\{u \in V_i \mid \mathsf{l}_{\mathsf{type}}(u) = \mathsf{h} \wedge u \notin \widehat{\mathcal{H}}_i\right\} ,$$
$$\widehat{\mathcal{A}}_i \triangleq \left\{u \in V_i \mid \mathsf{l}_{\mathsf{type}}(u) = \mathsf{a} \wedge \left[\exists v \in \widehat{\mathcal{H}}_i\colon u \text{ descendant of } v\right]\right\} ,$$
$$\mathcal{A}_i \triangleq \left\{u \in V_i \mid \mathsf{l}_{\mathsf{type}}(u) = \mathsf{a} \wedge u \notin \widehat{\mathcal{A}}_i\right\} .$$

Note that $\widehat{\mathcal{H}}_0 = \widehat{\mathcal{A}}_0 = \mathcal{A}_0 = \widehat{\mathcal{H}}_n = \widehat{\mathcal{A}}_n = \widehat{\mathcal{H}}_{n+1} = \mathcal{H}_{n+1} = \widehat{\mathcal{A}}_{n+1} = \mathcal{A}_{n+1} = \emptyset$ and $\mathcal{H}_0$ contains exactly the root vertex. Intuitively, $\widehat{\mathcal{H}}$ and $\widehat{\mathcal{A}}$ contain vertices that will need to be deferred in order to ensure that violating pairs are suitably serialized. It will be convenient to also define $\widehat{\mathcal{H}} = \bigcup_{i \in [n] \cup \{0\}} \widehat{\mathcal{H}}_i$ and analogously for $\mathcal{H}$, $\widehat{\mathcal{A}}$ and $\mathcal{A}$. Moreover, for each $i \in [n]$ let $\overline{a}_i \triangleq a_i - |\mathcal{A}_i| - |\widehat{\mathcal{A}}_i|$ and define $\overline{a}_0 = \overline{a}_{n+1} = 0$, intuitively $\overline{a}_i$ represents the number of adversarial successes in slot i that are left unused in F, i.e., do not have a corresponding vertex. Letting $h_i \triangleq |\widehat{\mathcal{H}}_i \cup \mathcal{H}_i|$, observe that $r \triangleq ((h_1, a_1), \ldots, (h_n, a_n))$ is a realization of w.

We now define w' along with its realization r'. For each $i \in [n+1]$ we define

$$a'_i \triangleq |\mathcal{A}_i| + |\widehat{\mathcal{A}}_{i-1}| + \overline{a}_{i-1} \quad , \quad h'_i \triangleq |\widehat{\mathcal{H}}_{i-1}| + |\mathcal{H}_i| \tag{3}$$

and $s'_i \triangleq \mathsf{round}_{\mathsf{H}}(h'_i)$, and we let

$$r' \triangleq ((h'_1, a'_1), \ldots, (h'_n, a'_n), (h'_{n+1}, a'_{n+1})) \in (\mathbb{N} \times \mathbb{N})^{n+1}$$
$$w' \triangleq ((s'_1, a'_1), \ldots, (s'_n, a'_n), (s'_{n+1}, a'_{n+1})) \in \Sigma_\infty^{n+1} ,$$

clearly r' is a realization of w'.

We now observe that $w' \in \mathcal{D}_1(w)$, as witnessed by realizations r and r'. Condition 1 of Definition 4.1 follows by simple accounting, it is sufficient to observe that for each $i \in [n+1]$ we have $a_i = |\mathcal{A}_i| + |\widehat{\mathcal{A}}_i| + \overline{a}_i$ and $a'_i = |\mathcal{A}_i| + |\widehat{\mathcal{A}}_{i-1}| + \overline{a}_{i-1}$ and moreover $|\widehat{\mathcal{A}}_0| = \overline{a}_0 = |\widehat{\mathcal{A}}_{n+1}| = |\mathcal{A}_{n+1}| = \overline{a}_{n+1} = 0$. Condition 2 holds by a similar argument.

The fork F' is then constructed as follows:

- the set of vertices, edges and the mapping $\mathsf{l}_{\mathsf{type}}$ of F' are identical to F (hence ensuring $F' \equiv F$);
- for any $u \in \mathcal{H}_i \cup \mathcal{A}_i$ we let $\mathsf{l}^{F'}_{\#}(u) := i$;
- for any $u \in \widehat{\mathcal{H}}_i \cup \widehat{\mathcal{A}}_i$ we let $\mathsf{l}^{F'}_{\#}(u) := i + 1$.

It remains to argue that F' is a valid fork and $F' \vdash_0 w'$. Axiom (A1) is trivially satisfied by construction: we know that the root is contained in $\mathcal{H}_0$ and hence its $\mathsf{l}^{F'}_{\#}$-label is 0.

Axiom (A2) is also preserved from F. To see this, note that by construction, if the $\mathsf{l}_{\#}(\cdot)$-label of any vertex u changes from F to F' then it increases by exactly 1. Moreover, we also claim that for any such honest vertex u whose $\mathsf{l}_{\#}(\cdot)$-label was increased and for any descendant vertex v of u having $\mathsf{l}_{\#}(v) = \mathsf{l}_{\#}(u)$, the $\mathsf{l}_{\#}(\cdot)$-label of v will also increase by 1 from F to F'. For adversarial v the claim follows directly from the definitions of $\widehat{\mathcal{A}}_i$ and $\mathsf{l}^{F'}_{\#}(\cdot)$; while for honest v, this is because if u is a first coordinate of some violating pair (u, w) then (v, w) must also be a violating pair and hence the claim follows from the definitions of $\widehat{\mathcal{H}}_i$ and $\mathsf{l}^{F'}_{\#}(\cdot)$. To conclude, if the $\mathsf{l}_{\#}(\cdot)$-label of a vertex gets changed from F to F', then it increases by 1 and all the $\mathsf{l}_{\#}(\cdot)$-labels of its children having the same label increase as

well, therefore the weak monotonicity of the $\mathsf{l}_\#(\cdot)$-labeling along any tine remains satisfied.

Axiom (A3) can be verified for F' by simple accounting. For adversarial vertices u, the label $\mathsf{l}_\#(u) = i$ is by construction of $\mathsf{l}^{F'}_\#$ attributed exactly to all vertices in $\widehat{\mathcal{A}}_{i-1} \cup \mathcal{A}_i$, and this is aligned with the definition of a_i in (3). The argument for honest vertices is analogous; it just needs to additionally take into account the "coarser" accounting of honest vertices using symbols $\{0, \mathsf{h}, \mathsf{H}\}$.

Finally we verify axiom (A4) for F' and $\Delta = 0$. Towards a contradiction, assume there exist honest vertices u, v in F' such that $\mathsf{l}^{F'}_\#(u) < \mathsf{l}^{F'}_\#(v)$ and $\mathsf{len}_{F'}(u) \geq \mathsf{len}_{F'}(v)$ (note that $\mathsf{len}(\cdot)$ does not change from F to F' so we omit the subscript from now on). Consider two cases:

$\mathsf{l}^F_\#(u) < \mathsf{l}^F_\#(v)$: This means that $(u, v) \in \mathcal{V}(F)$ and hence $\mathsf{l}^{F'}_\#(u) = \mathsf{l}^F_\#(u)+1$ by construction. Since $\mathsf{len}(u) \geq \mathsf{len}(v)$ and $F \vdash_1 w$, we have $\mathsf{l}^F_\#(v) \leq \mathsf{l}^F_\#(u) + 1$. At the same time $\mathsf{l}^{F'}_\#(u) < \mathsf{l}^{F'}_\#(v)$ and the $\mathsf{l}_\#$-labels change by at most 1 from F to F', therefore we must have $\mathsf{l}^{F'}_\#(v) = \mathsf{l}^F_\#(v) + 1$ and $\mathsf{l}^F_\#(v) = \mathsf{l}^F_\#(u) + 1$. This in turn implies $v \in \widehat{\mathcal{H}}$ and that means that there exists a vertex $w \in F$ such that $(v, w) \in \mathcal{V}(F)$. However, the pair (u, w) now violates axiom (A4) for $\Delta = 1$ in F as we have $\mathsf{l}^F_\#(u) < \mathsf{l}^F_\#(v) < \mathsf{l}^F_\#(w)$ and $\mathsf{len}(u) \geq \mathsf{len}(v) \geq \mathsf{len}(w)$, which is a contradiction.

$\mathsf{l}^F_\#(u) \geq \mathsf{l}^F_\#(v)$: Given that $\mathsf{l}^{F'}_\#(u) < \mathsf{l}^{F'}_\#(v)$ and the $\mathsf{l}_\#$-labels change by at most 1 from F to F', we must have $\mathsf{l}^F_\#(u) = \mathsf{l}^F_\#(v)$, $v \in \widehat{\mathcal{H}}$ and $u \in \mathcal{H}$. In particular, $v \in \widehat{\mathcal{H}}$ implies the existence of a vertex w such that $(v, w) \in \mathcal{V}(F)$, i.e., $\mathsf{l}^F_\#(v) < \mathsf{l}^F_\#(w)$ and $\mathsf{len}(v) \geq \mathsf{len}(w)$. However, this gives us $\mathsf{len}(u) \geq \mathsf{len}(v) \geq \mathsf{len}(w)$ and $\mathsf{l}^F_\#(u) = \mathsf{l}^F_\#(v) < \mathsf{l}^F_\#(w)$, meaning that $(u, w) \in \mathcal{V}(F)$ and hence $u \in \widehat{\mathcal{H}}$, a contradiction.

This concludes the argument that $F' \vdash_0 w'$ and the proof of the lemma. □

We can now establish the following lemma, which is again an analogue of Corollary 3.5, and is proven in Appendix B.

Lemma 4.3. *Let $w \in \Sigma^n_\infty$, then $\beta^1_\ell(w) \leq \max_{w' \in \mathcal{D}_1(w)} \beta^0_{\ell+1}(w') + 2$.*

4.2 The Recurrence $\mathbf{B}_\ell(\cdot)$

In this section we define an easily computable recurrent function $\mathbf{B}_\ell$ that we later use to upper-bound β_ℓ of a particular string w. The definition of $\mathbf{B}_\ell$ will be composed of several basic functions that we define first. After that, we give a recursive description of how $\mathbf{B}_\ell$ can be computed using these basic constituent operations.

The basic intuition underlying the computation of $\mathbf{B}_\ell(w)$ is to internally simulate the computation of $\beta^0_\ell(w')$ on all possible deferrals $w' \in \mathcal{D}_1(w)$, as that is precisely described in Theorem 3.6.

More concretely, $\mathbf{B}_\ell$ returns a tuple

$$\mathbf{B}_\ell(w) = ((\beta_0, a_0), (\beta_\mathsf{H}, a_\mathsf{H}), (\beta_\mathsf{h}, a_\mathsf{h})) \in (\mathbb{Z} \times \mathbb{N})^3$$

where each pair (β_s, a_s) for $s \in \{0, \mathsf{H}, \mathsf{h}\}$ keeps track of the best (in a well-defined sense detailed below) achievable margin β_s and the number of delayed adversarial successes a_s after processing a deferral of w that: (0) does not produce an honest carry-over from slot $|w|$ to $|w| + 1$; (h) produces a single such honest carry-over; or (H) produces a multi-honest such carry-over. The definition of $\mathbf{B}_\ell$ then describes how to update this tuple $\mathbf{B}_\ell(w)$ to arrive at $\mathbf{B}_\ell(wz)$ for any $z \in \Sigma_\infty$.

Basic operations. For any $(\beta, a, a') \in \mathbb{Z} \times \mathbb{N} \times \mathbb{N}$ we introduce the following functions:

$$\begin{aligned} \mathsf{NHE}(\beta, a, a') &\triangleq (\beta + a, a') \,, \\ \mathsf{HE}(\beta, a, a') &\triangleq (\beta + a - 1, a') \,, \\ \mathsf{NO}(\beta, a, a') &\triangleq \begin{cases} (\max\{0, \beta + a\}, a' + \min\{0, \beta + a\}) \\ \qquad\qquad \text{if } \beta \in \{-a - a', \ldots, 0\}, \\ \mathsf{HE}(\beta, a, a') \qquad \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Their names stand for *(no) honest effect* and *neutralization opportunity*, respectively. Intuitively, these functions will be invoked in the update step computing $\mathbf{B}_\ell(wz)$ from $\mathbf{B}_\ell(w)$ with their inputs (β, a) being one of the pairs (β_s, a_s) in $\mathbf{B}_\ell(w)$ for some s, and a' being the number of adversarial successes in the currently processed symbol z. The functions then return a new, updated value pair (β^*, a^*) if (NHE) there was no honest effect on β_ℓ in this round (e.g., no delayed honest success from previous slot and no honest success in this slot either); or (HE) there was an effect of a honest success that decreased β_ℓ by 1; or (NO) there was a neutralization opportunity and whether an honest effect occurred depends on the current running value of β.

Note that which of these basic functions are invoked when computing $\mathbf{B}_\ell(wz)$ from $\mathbf{B}_\ell(w)$ depends on information external to these functions: the honest carry from previous slot (i.e., which s is used to index into the previous tuple $\mathbf{B}_\ell(w)$), the honest success(es) recorded in the current symbol z, and the desired honest carry to the next slot (i.e., which pair of the new value $\mathbf{B}_\ell(wz)$ is being computed). In all cases, these functions are chosen to match the behavior of β^0_ℓ on the respective deferral as described by Theorem 3.6. Looking ahead, this inductive property will be established in Lemma 4.5.

For notational convenience, we also introduce a function HE^t_ℓ that behaves as NO or HE depending on two parameters $\ell, t \in \mathbb{N}$; ℓ will be the usual parameter of β_ℓ and t will be the current slot—HE^t_ℓ will hence be used to distinguish the "pre-ℓ" and "post-ℓ" settings:

$$\mathsf{HE}^t_\ell(\beta, a, a') \triangleq \begin{cases} \mathsf{NO}(\beta, a, a') \text{ if } t < \ell, \\ \mathsf{HE}(\beta, a, a') \text{ if } t \geq \ell. \end{cases}$$

To reason about these basic functions, we introduce a binary relation $\preceq$ on the elements $(\beta, a) \in \mathbb{Z} \times \mathbb{N}$ as follows:

$$\begin{aligned} (\beta_1, a_1) \preceq (\beta_2, a_2) :\Leftrightarrow [&(\beta_1 + a_1 < \beta_2 + a_2) \vee \\ &\vee (\beta_1 + a_1 = \beta_2 + a_2 \wedge a_1 \leq a_2)] \,. \end{aligned} \tag{5}$$

It is easy to verify that $\preceq$ is in fact a total order on $\mathbb{Z} \times \mathbb{N}$. We use the standard notation $x \prec y$ for $(x \preceq y \wedge x \neq y)$. For convenience, let us define an operator $\max_\prec$ that, given a tuple $\{(x_i, y_i)\}_{i=1}^n$ of pairs from $\mathbb{Z} \times \mathbb{N}$, returns the maximum pair with respect to the total order $\preceq$. Finally, let $\perp$ represent the pair $(-\infty, 0)$; to handle $\perp$ we sometimes abuse the notation and extend $\preceq$ to $(\mathbb{Z} \cup \{-\infty\}) \times \mathbb{N}$ in the natural way. We also sometimes treat $\perp$ as a ternary function (akin to NHE, HE, NO) that always returns $(-\infty, 0)$, which will always be clear from the context.

Formal description of $\mathbf{B}_\ell$. Let $\mathbf{B}_\ell(\varepsilon) \triangleq ((0,0), \perp, \perp)$. Furthermore, if $\mathbf{B}_\ell(w) = ((\beta_0, a_0), (\beta_\mathsf{H}, a_\mathsf{H}), (\beta_\mathsf{h}, a_\mathsf{h}))$ and $|w| + 1 = t$ then

$$\mathbf{B}_\ell(w(0,a')) = (\max_\prec \left\{\begin{matrix}\mathsf{NHE}(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{HE}^t_\ell(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}, \perp, \perp) \,,$$

$$\mathbf{B}_\ell(w(\mathsf{H},a')) = (\max_\prec \left\{\begin{matrix}\mathsf{NO}(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{NO}(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}, \max_\prec \left\{\begin{matrix}\mathsf{NHE}(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{NO}(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}, \max_\prec \left\{\begin{matrix}\mathsf{NO}(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{NO}(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}) \,,$$

$$\mathbf{B}_\ell(w(\mathsf{h},a')) = (\max_\prec \left\{\begin{matrix}\mathsf{HE}^t_\ell(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{NO}(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}, \perp, \max_\prec \left\{\begin{matrix}\mathsf{NHE}(\beta_0, a_0, a') \\ \mathsf{NO}(\beta_\mathsf{H}, a_\mathsf{H}, a') \\ \mathsf{HE}^t_\ell(\beta_\mathsf{h}, a_\mathsf{h}, a')\end{matrix}\right\}) \,.$$

We additionally introduce some notation and terminology that allows us to conveniently reason about $\mathbf{B}_\ell$. For some $\mathbf{B}_\ell(w) = ((\beta_0, a_0), (\beta_\mathsf{H}, a_\mathsf{H}), (\beta_\mathsf{h}, a_\mathsf{h}))$ and $s \in \{0, \mathsf{H}, \mathsf{h}\}$ we use the notation $\mathbf{B}_\ell(w)[s]$ to refer to the pair (β_s, a_s) in $\mathbf{B}_\ell(w)$. Moreover, we let

$$B_\ell(w) \triangleq \max_{s \in \{0,\mathsf{H},\mathsf{h}\}} \beta_s + a_s \,.$$

Intuitively, given some $w \in \Sigma^*_\infty$ and $z \in \Sigma_\infty$, the final step of computation of $\mathbf{B}_\ell(wz)$ (processing the trailing symbol $z \in \Sigma_\infty$) can be seen as determined by a three-dimensional table of $3^3 = 27$ cells, each cell specifying a single operation $\mathsf{op} \in \{\mathsf{NHE}, \mathsf{HE}^t_\ell, \mathsf{NO}, \perp\}$ that needs to be applied to $\mathbf{B}_\ell(w)[s_\mathsf{prev}]$ if the "honest carry" from the previous step is s_prev, the honest part of the current symbol z is s_cur (i.e., $z = (s_\mathsf{cur}, a')$), and the desired honest carry to the next slot is s_next; with all $s_\mathsf{prev}, s_\mathsf{cur}, s_\mathsf{next} \in \{0, \mathsf{H}, \mathsf{h}\}$. We sometimes explicitly refer to this operation as $\mathsf{op}_{[s_\mathsf{prev}, s_\mathsf{cur}, s_\mathsf{next}]} \in \{\mathsf{NHE}, \mathsf{HE}^t_\ell, \mathsf{NO}, \perp\}$. For example $\mathsf{op}_{[0,0,0]} \equiv \mathsf{NHE}$, $\mathsf{op}_{[\mathsf{H},0,0]} \equiv \mathsf{NO}$, $\mathsf{op}_{[\mathsf{h},0,0]} \equiv \mathsf{HE}^t_\ell$, $\mathsf{op}_{[0,0,s]} \equiv \perp$ for any $s \in \{0, \mathsf{h}, \mathsf{H}\}$, and so on.

Monotonicity. We conclude this section by stating a simple monotonicity property of all the basic functions NHE, HE, NO and HE^t_ℓ underlying $\mathbf{B}_\ell$. Given partial orders $(S, \prec_S)$ and $(T, \prec_T)$, recall that a function $f\colon S \to T$ is called (weakly) monotone if

$$\forall x, y \in S\colon\ (x \preceq_S y \Rightarrow f(x) \preceq_T f(y)) \,.$$

We defer the proof of the following lemma to Appendix B.

Lemma 4.4. *For any fixed $a' \in \mathbb{N}$ and $t, \ell \geq 1$, the functions $\mathsf{NHE}(\cdot,\cdot,a')$, $\mathsf{HE}(\cdot,\cdot,a')$, $\mathsf{HE}^t_\ell(\cdot,\cdot,a')$ and $\mathsf{NO}(\cdot,\cdot,a')$ mapping $\mathbb{Z} \times \mathbb{N} \to \mathbb{Z} \times \mathbb{N}$ are monotone with respect to the total order $\preceq$ of* (5).

4.3 Upper-bounding Deferral Margin by $\mathbf{B}_\ell(\cdot)$

The following lemma is the key technical result that formalizes the intuition behind the definition of $\mathbf{B}_\ell$. We provide an outline of its proof below, deferring the full details to Appendix B.

Lemma 4.5. *Let $w \in \Sigma^n_\infty$ and let $w' \in \mathcal{D}_1(w)$. Writing $w' = x'(s'_{n+1}, a'_{n+1})$, so that $x' \in \Sigma^n_\infty$ consists of the first n symbols of w' and $(s'_{n+1}, a'_{n+1}) \in \{0, \mathsf{H}, \mathsf{h}\} \times \mathbb{N}$ is the last symbol. Then we have*

$$\left(\beta^0_\ell(x'), a'_{n+1}\right) \preceq \mathbf{B}_\ell(w)[s'_{n+1}] \,.$$

Proof. We proceed by induction on the length $n \in \mathbb{N}$ of $w \in \Sigma^*_\infty$.

Base case. With $n = 0$, we have $w = \varepsilon$, $w' \in \mathcal{D}_1(w) = \{(0,0)\}$, $\mathbf{B}_\ell(w) = ((0,0), \perp, \perp)$ and hence $w' = x'(s'_{n+1}, a'_{n+1})$ for $x' = \varepsilon$ and $s'_{n+1} = a'_{n+1} = 0$. This implies $(\beta^0_\ell(x'), a'_{n+1}) = (\beta^0_\ell(\varepsilon), 0) = (0,0) = \mathbf{B}_\ell(w)[0]$, as desired.

Induction step. Let $w \in \Sigma^n_\infty$ and $w' \in \mathcal{D}_1(w)$. Write $w = x(s_n, a_n)$, where x consists of the first $n-1$ symbols of w. We will first construct $\overline{x}'$ that is a deferral of x and shares the first $n-1$ symbols as x'. We observe that w' naturally gives rise to a deferral of x. To describe this, let $r = (h_1, a_1), \ldots, (h_n, a_n)$ and $r' = (h'_1, a'_1), \ldots, (h'_{n+1}, a'_{n+1})$ be realizations of w and w', respectively, for which r' is a deferral of r. Letting q denote the first $n-1$ symbols of the realization r, it's clear that q is a realization of x. Then we observe that an adaptation of the suffix of r' (and w') yields a deferral of x (the prefix of w). Specifically, defining

$$(\overline{h}'_n, \overline{a}'_n) = (h'_n, a'_n) + (h'_{n+1}, a'_{n+1}) - (h_n, a_n)$$

(where arithmetic is coordinatewise) it is easy to confirm that $\overline{q}' \triangleq (h'_1, a'_1), \ldots, (h'_{n-1}, a'_{n-1}), (\overline{h}'_n, \overline{a}'_n)$ is a deferral of the realization q of x. To see this, observe that by construction,

$$\overline{h}'_n + \sum_{i=1}^{n-1} h'_i = \left(\sum_{i=1}^{n+1} h'_i\right) - h_n = \sum_{i=1}^{n-1} h_i$$

and

$$\overline{a}'_n + \sum_{i=1}^{n-1} a'_i = \left(\sum_{i=1}^{n+1} a'_i\right) - a_n = \sum_{i=1}^{n-1} a_i \,,$$

so the full sums are correct and the remaining nested sums (items (1) and (2) of Definition 4.1) follow from the fact that r' is a deferral of r. To reiterate and organize the notation, we arrange these in a table, where we use the notation $w \leftarrow r$ to indicate that r is a realization of the string w, and $w \rightsquigarrow w'$ to indicate that w' is a 1-deferral of w.

$$\begin{array}{l} w \leftarrow r = (h_1, a_1), \ldots, (h_n, a_n) \\ \wr \\ w' \leftarrow r' = (h'_1, a'_1), \ldots, (h'_n, a'_n), (h'_{n+1}, a'_{n+1}) \end{array}$$

$$\begin{array}{l} x \leftarrow q = (h_1, a_1), \ldots, (h_{n-1}, a_{n-1}) \\ \wr \\ \overline{x}' \leftarrow \overline{q}' = (h'_1, a'_1), \ldots, (h'_{n-1}, a'_{n-1}), (\overline{h}'_n, \overline{a}'_n) \end{array}$$

Let z' be the $(n-1)$-prefix of w' (or x') and let $\overline{s}'_n \in \{0, \mathsf{h}, \mathsf{H}\}$ be the "rounded" version of $\overline{h}'_n$, i.e., $\overline{s}'_n \triangleq \mathsf{round}_\mathsf{H}(\overline{h}'_n)$. By induction hypothesis we have

$$(\beta^0_\ell(z'), \overline{a}'_n) \preceq \mathbf{B}_\ell(x)[\overline{s}'_n] \,. \tag{6}$$

The inductive step of the argument is now established in a sequence of manipulations that respect the ordering $\succeq$. We first give an overview of this sequence and then justify each of its steps in

detail. Namely, we will prove that

$$\begin{aligned}\mathbf{B}_\ell(w)[s'_{n+1}] &\overset{(a)}{=} \max_{\prec}\left\{\begin{matrix}\mathrm{op}_{[0,s_n,s'_{n+1}]}\left(\mathbf{B}_\ell(x)[0],a_n\right)\\ \mathrm{op}_{[\mathsf{H},s_n,s'_{n+1}]}\left(\mathbf{B}_\ell(x)[\mathsf{H}],a_n\right)\\ \mathrm{op}_{[\mathsf{h},s_n,s'_{n+1}]}\left(\mathbf{B}_\ell(x)[\mathsf{h}],a_n\right)\end{matrix}\right\}\\ &\overset{(b)}{\succeq} \mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]}\left(\mathbf{B}_\ell(x)\left[\overline{s}'_n\right],a_n\right) \qquad (7)\\ &\overset{(c)}{\succeq} \mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]}\left(\beta^0_\ell(z'),\overline{a}'_n,a_n\right)\\ &\overset{(d)}{=} \left(\beta^0_\ell(x^*),a^*\right) \overset{(e)}{\succeq} \left(\beta^0_\ell(x'),a'_{n+1}\right),\end{aligned}$$

where x^*, a^* are simple modifications of x', a'_{n+1} that we precisely define below. Note that establishing (7) concludes the inductive step and hence also the whole proof of the lemma.

Equation (a) follows from the definition of $\mathbf{B}_\ell$, recall that $\mathrm{op}_{[s,s_n,s'_{n+1}]}$ is the operation that is used in the computation of $\mathbf{B}_\ell$ in the cell where the honest carry from previous slot is s, the honest part of the symbol in the current slot is s_n, and the desired honest carry to the next slot is s'_{n+1}. Step (b) then follows by definition of $\max_{\prec}$. Step (c) is a direct application of the induction hypothesis (6) and the monotonicity of $\mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]}$ with respect to its first two inputs, as established in Lemma 4.4.

Towards justifying step (d), we first define x^* and a^*. Let $w^* \in \Sigma_\infty^{n+1}$ be a 1-deferral of w that is identical to w' except for its last two symbols, and, intuitively, these two symbols only differ from the respective symbols in w' by a potentially different number of adversarial successes being deferred from slot n to slot $n+1$. Formally, if $r' = (h'_1, a'_1), \ldots, (h'_n, a'_n), (h'_{n+1}, a'_{n+1})$ is a realization of w', then w^* corresponds to a realization

$$w^* \leftarrow r^* = (h'_1, a'_1), \ldots, (h'_n, a'_n + a'_{n+1} - a^*), (h'_{n+1}, a^*)$$

where

$$a^* = \begin{cases} a_n & \text{if } \mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]} \in \{\mathsf{NHE}, \mathsf{HE}^n_\ell\}, \\ a_n + \min\left\{0, \beta^0_\ell(z') + \overline{a}'_n\right\} & \text{if } \mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]} \equiv \mathsf{NO}. \end{cases}$$

We let x^* denote the first n symbols of w^* (just like x' is related to w'), i.e., $w^* = x^*(s'_{n+1}, a^*)$; and for convenience let x^*_n be the last symbol of x^*.

Observe that, intuitively, in all three cases ($\mathsf{NHE}, \mathsf{HE}^n_\ell, \mathsf{NO}$), the value a^* is defined to be exactly the number of adversarial successes that are deferred by the respective operation (i.e., the second coordinate of its output) according to its definition (4), given input $(\beta^0_\ell(z'), \overline{a}'_n, a_n)$. Note that the fact that the second component of $\mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]}(\beta^0_\ell(z'), \overline{a}'_n, a_n)$ equals a^* (which is a part of step (d)) follows from the definition of a^*.

The main effort in establishing the induction case lies in verifying the other part of step (d), namely, the first component of $\mathrm{op}_{[\overline{s}_n,s_n,s'_{n+1}]}\left(\beta^0_\ell(z'), \overline{a}'_n, a_n\right)$ being equal to $\beta^0_\ell(x^*)$. This amounts to verifying that, intuitively, the operation performed in the cell of the definition of $\mathbf{B}_\ell$ determined by $(\overline{s}_n, s_n, s'_{n+1})$ is identical to how $\beta^0_\ell(x^*) = \beta^0_\ell(z'x^*_n)$ evolves from $\beta^0_\ell(z')$ when processing the last symbol x^*_n of x^*. Luckily, this behavior of β^0_ℓ is exactly described by Theorem 3.6, and hence this claim can be verified by a straightforward case analysis considering each of the cells separately and comparing it to the behavior guaranteed by Theorem 3.6.

For illustration, consider the three cells involved in the case when $s_n = s'_{n+1} = 0$, and hence according to the definition of $\mathbf{B}_\ell$, we have

$$\mathbf{B}_\ell(x(0,a_n))[0] = \max_{\prec}\left\{\begin{matrix}\mathsf{NHE}(\beta_0,a_0,a_n)\\ \mathsf{NO}(\beta_\mathsf{H},a_\mathsf{H},a_n)\\ \mathsf{HE}^n_\ell(\beta_\mathsf{h},a_\mathsf{h},a_n)\end{matrix}\right\}, \qquad (8)$$

where the values $((\beta_0,a_0),(\beta_\mathsf{H},a_\mathsf{H}),(\beta_\mathsf{h},a_\mathsf{h}))$ are taken from $\mathbf{B}_\ell(x)$. The intuitive way to read the above equation (8) is that the new value $\mathbf{B}_\ell(x(s_n,a_n))[0]$ after processing (s_n,a_n) will be computed as a maximum of three possible evolutions: either

(i) starting from (β_0,a_0) (representing no deferred honest success from the previous slot $n-1$ to the current slot n in x^*) and applying NHE;
(ii) starting from $(\beta_\mathsf{H},a_\mathsf{H})$ (representing multiple such deferred honest successes) and applying NO; or
(iii) starting from $(\beta_\mathsf{h},a_\mathsf{h})$ (representing a single such deferred honest success) and applying HE^n_ℓ.

Observe that this is, according to Theorem 3.6, exactly the behavior of β^0_ℓ on the last symbol of x^*:

(i) If there are no deferred honest successes from the previous slot, given that there are also no honest successes in this slot ($s_n = 0$), β^0_ℓ simply increases by a_0 and $a^* = a_n$ adversarial successes are deferred to the next slot, i.e., NHE is applied.
(ii) If multiple honest successes were deferred from the previous slot, β^0_ℓ again increases by a_0 but also potentially decreases by 1 to account for the carried-over honest successes, unless β_H is in the appropriate range that allows for "neutralizing" this effect. If a neutralization is possible, the number of carried-over adversarial successes a^* is chosen to be maximal while ensuring the neutralization happens. I.e., NO is applied.
(iii) If a single honest success was deferred from the previous slot, β^0_ℓ again increases by a_0, but if we are in a slot $n > \ell$ then is also guaranteed to decrease by 1 to account for processing the carried-over honest success. Given that there are no honest successes in this slot ($s_n = 0$, a single honest success (h) cannot be neutralized after slot ℓ. On the other hand, in the case $n < \ell$, β^0_ℓ is only decreased by 1 if it is positive, again in agreement with Theorem 3.6. I.e., overall, HE^n_ℓ is applied.

The reasoning for all other cells of $\mathbf{B}_\ell$ is fully analogous.

Finally, to justify step (e), we need to argue that, all other things equal, deferring any different number of adversarial successes than a^* in the last slot of w will lead to a pair $(\beta', a') \prec (\beta^* = \beta^0_\ell(x^*), a^*)$. Observe that we can argue this separately for each of the cases where β^0_ℓ behaves according to NHE, HE, and NO respectively. In light of the analysis above, β^0_ℓ always follows one of these operations and the choice of it depends only on the pattern of deferred honest successes and the length n, which are identical in w^* and w'. For NHE and HE, the desired property is immediate, as w^* defers all available adversarial successes in these cases; deferring fewer of them (i.e., choosing $a' < a^*$) would result in a pair (β', a') such that $\beta' + a' = \beta^* + a^*$ but $a' < a^*$, and hence $(\beta', a') \prec (\beta^*, a^*)$. For NO, deferring a smaller number of adversarial successes (i.e., choosing $a' < a^*$) would have the same effect, i.e., it would lead to (β', a') such that $\beta' + a' = \beta^* + a^*$ but $a' < a^*$. On the other hand, choosing $a' > a^*$ would prevent "neutralizing" the honest success in slot n, leading to a pair (β', a') such that $\beta' + a' = \beta^* + a^* - 1$,

again implying $(\beta', a') \prec (\beta^*, a^*)$ as desired. This concludes the justification of step (e) and hence the whole proof. □

Given Lemma 4.5, we can now establish our main result.

Theorem 4.6. *Let $w \in \Sigma_\infty^*$. Then*

$$\beta_\ell^1(w) \le B_{\ell+1}(w) + 2 \ .$$

Proof. First, Lemma 4.3 gives us

$$\beta_\ell^1(w) \le \max_{w' \in \mathcal{D}_1(w)} \beta_{\ell+1}^0(w') + 2 \ .$$

Let $w^* \in \mathcal{D}_1(w)$ be the 1-deferral of w that maximizes $\beta_{\ell+1}^0(\cdot)$ above, and as before let $w^* = x^*(s_{n+1}^*, a_{n+1}^*)$ with $x^* \in \Sigma_\infty^n$ and $(s_{n+1}^*, a_{n+1}^*) \in \{0, \mathsf{H}, \mathsf{h}\} \times \mathbb{N}$. Let $\mathbf{B}_{\ell+1}(w) = ((\beta_0, a_0), (\beta_\mathsf{H}, a_\mathsf{H}), (\beta_\mathsf{h}, a_\mathsf{h}))$, then we have

$$\beta_{\ell+1}^0(w^*) \overset{\text{(a)}}{\le} \beta_{\ell+1}^0(x^*) + a_{n+1}^* \overset{\text{(b)}}{\le} \max_{s \in \{0,\mathsf{H},\mathsf{h}\}} \beta_s + a_s = B_{\ell+1}(w)$$

as desired, where inequality (a) follows from Theorem 3.6, and inequality (b) is a direct consequence of Lemma 4.5. □

Finally, we remark that for characteristic strings of the special form $w = w'(0,0)$, i.e. terminating with a success-free slot, we clearly have $\beta_\ell^1(w) = \beta_\ell^0(w)$, this leads to a stronger statement without the additional additive term +2 for this special case.

5 EXPLICIT BOUNDS

Finally, we study explicit bounds provided by this analysis. As described, we are interested in the setting where honest and adversarial block production are determined by Poisson processes with parameters r_h and r_a, while network delivery of blocks may be delayed by Δ_r time units. This induces the distribution on the characteristic string w where each symbol $w_i = (s_i, a_i) \in \{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N}$ is independent and: (i) a_i is Poisson-distributed with parameter $r_a\Delta_\mathrm{r}$, and (ii) s_i is determined by a Poisson random variable X with parameter $r_h\Delta_\mathrm{r}$ so that $s_i = \mathsf{round}_\mathsf{H}(X)$ (cf. (1)). Let $D(r_a, r_h, \Delta_\mathrm{r}; n)$ denote the distribution on $(\{0, \mathsf{h}, \mathsf{H}\} \times \mathbb{N})^n$ given by this rule.

We collect results for both a Bitcoin-like system—with 600 second inter-block periods corresponding to a 1/600 rate Poisson process—and an Ethereum-like system—with 13 second inter-block periods corresponding to a 1/13 rate process. The 90th percentile block propagation time for Bitcoin (resp. Ethereum) has been measured to be around 4 seconds [16] (resp. around 2 seconds [6], partly due to smaller block sizes); we will use these values as the typical delays in the respective cases. To provide more data that are directly comparable with a previous work [15], we will also give results for a 10 seconds delay for Bitcoin and a 5 seconds delay for Ethereum.

Temporal settlement rules. Examining the conclusions of the previus section and, in particular, the recursive description of the tuple $\mathbf{B}_\ell$, it is clear that one can efficiently determine the value $\mathbf{B}_\ell(w)$ for any particular characteristic string w. Furthermore, considering that the distribution $D(r_a, r_h, \Delta_\mathrm{r}; n)$ calls for independent symbols, it is a straightforward matter to determine the exact distribution of $\mathbf{B}_\ell(wa)$, where a is an additional independent symbol, from that of $\mathbf{B}_\ell(w)$. Specifically, we consider a "six-dimensional" table T_n, with one cell for each possible value of $\mathbf{B}_\ell$ (thus a value has the form $(\beta_0, a_0, \beta_\mathsf{h}, a_\mathsf{h}, \beta_\mathsf{H}, a_\mathsf{H})$), whose cells are populated with the probabilities that this value emerges in $\mathbf{B}_\ell(w)$ (with w drawn from $D(r_a, r_h, \Delta_\mathrm{r}; n)$). Given the "kernel" distribution for the next symbol a, each cell of the corresponding table T_{n+1} can be determined as an appropriate convex combination of the entries in T_n with the kernel distribution. Of course, the symbol distribution has infinite support; however, the Poisson distribution decays very rapidly so it is straightforward to use finite approximations that suitably control errors.

Initially, we must settle on a distribution of $\mathbf{B}_\ell$ at time ℓ (corresponding to the moment in time when the transaction of interest was submitted to the blockchain). While this does depend on ℓ, the distribution converges quickly to an exponentially decaying distribution (in the sense that the entries are $\exp(-\lambda(\beta_0 + a_0))$); for this reason, rather than select some particular ℓ in our experimental results, we choose a very large ℓ that corresponds to the steady state of the blockchain. Specifically, we select a large enough ℓ so that the difference in total variation observed by evolving for an additional step is bounded by 10^{-5}. (Intuitively, this initial distribution reflects the number of private blocks that the adversary may have, along with any deferred honest blocks from slot ℓ.)

For simplicity, we append a concluding $(0,0)$ onto the end of the generated characteristic string which, recalling the semantics of $\mathbf{B}_\ell$, permits us to focus on a single pair, (β_0, a_0); as the string does not terminate with any honest victories, we may neglect the $+2$ of Theorem 4.6 and the event of interest is simply $\beta_0 + a_0 \ge 0$, in which case the adversary can launch a successful "double spend" attack. Note that this postpended $(0,0)$ in fact corresponds to an observable event–it can be guaranteed by witnessing a "quiet" region of length $2\Delta_\mathrm{r}$. Finally, we are responsible for computing the probability that the margin should *ever* climb above zero after our threshold of interest. This we compute by continuing to evolve the probability forward in time, but effectively "freezing" any probability mass on positive values of margin. We then evolve the system forward until the (exponentially decaying) contributions from further evolution are negligible.

Figures 3 and 4 give an overview of our results for temporal settlement in Bitcoin and Ethereum, respectively. These figures depict both lower bounds and upper bounds on the settlement error for the considered blockchain as a function of time, for both considered values of network delays (denoted Δ_r), and are hence more detailed versions of Figures 1 and 2, respectively. The lower bounds shown in these figures are obtained using the lock-step analysis from Section 3 and assuming that neutralization opportunities never occur. Figure 1, in particular, clearly shows that our method obtains highly accurate settlement times for the temporal settlement rule for Bitcoin.

Towards comparing with prior art, as outlined in the introduction, the most relevant previous work is [15]. As an example, their method allows to conclude that for a 10% adversary and $\Delta_\mathrm{r} = 10s$, a Bitcoin block is secured with less than 10^{-3} error probability after 5 hours 20 minutes of confirmation time, while our new results conclude this after about 2 hours and 30 minutes. Furthermore, the new results are no more than 2 minutes and 30 seconds away from the optimum. The comparison is similarly favorable to our results for the Ethereum parametrization. For comparison, we provide also plots of the upper bounds from [15] in these figures. A

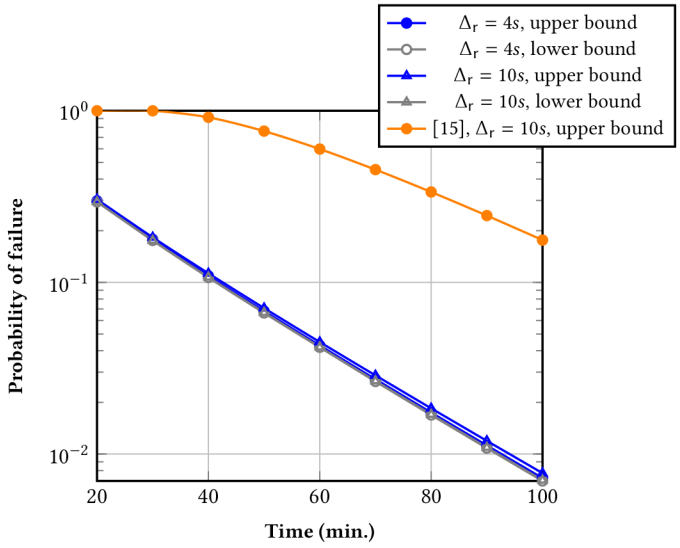

Figure 3: Bitcoin temporal settlement failure for a 10% adversary, results from [15] for comparison.

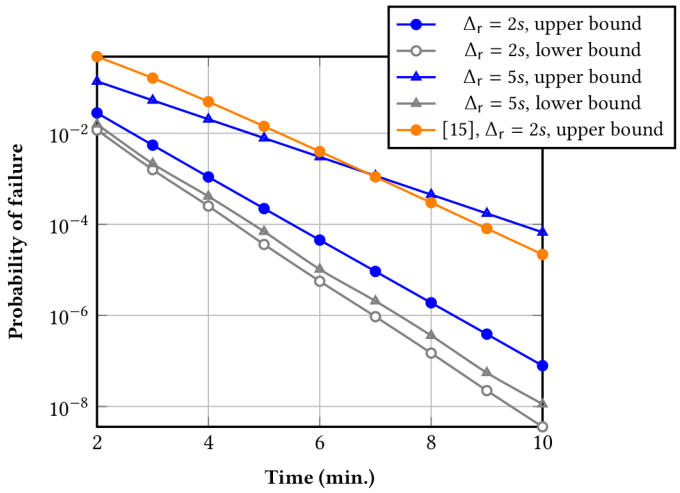

Figure 4: Ethereum temporal settlement failure for a 10% adversary, results from [15] for comparison.

more detailed record of our results for temporal settlement is given in Table 1 in Appendix C.

A final remark on the tightness claims of our time-based settlement results. Throughout the paper (including the previous paragraph) we mention that our upper bounds are a particular number of seconds away from optimal: e.g., for Bitcoin settlement at the one-hour mark with 10-second delays and a 10% adversary this is about 90 seconds. This refers to the facts at the 1 hour mark, the protocol settles except with probability 4.489% (given by the upper bound in the 1-deferral setting) , while 90 seconds before that, an adversary can prevent settlement with probability 4.494% (given by the lower bound in the 0-deferral setting). (Some of the results used

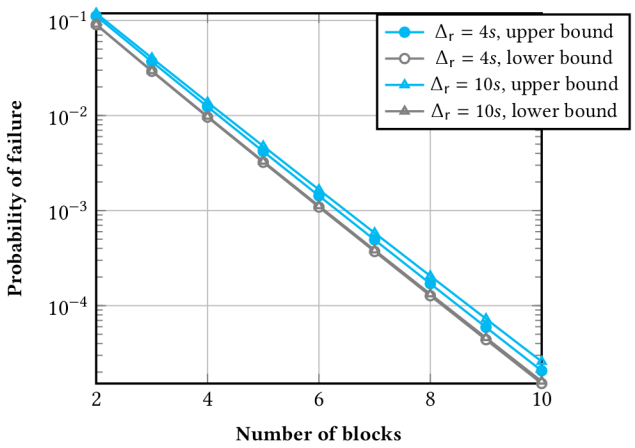

Figure 5: Bitcoin block-based settlement failure for a 10% adversary.

for comparisons to lower-bounds and prior work are not included in the tables of results below, but are obtained using exactly the same methods.)

Block-based settlement rules. We then transition to the question of settlement with a more sophisticated stopping condition: "Wait for the transaction to be buried by k blocks." This requires a small adaptation to the framework above because an individual symbol may generate multiple blocks: in this case, one maintains a graded data structure that reflects the probabilities conditioned on observing a particular total number of block-creation events. A further complication arises in the interpretation of margin for this stopping time. In particular, this stopping time is quite different from the simple stopping time "wait for k block creation events," which is not even an observable event. For example, note that if $\beta_\ell()$ is $2k$ at time ℓ, an adversary can immediately activate the stopping time "wait for the transaction to be buried by k blocks" and can, furthermore, double spend. An interesting, and quite powerful, feature of this stopping time is that it naturally adapts to adversarial behavior in the sense that withheld adversarial blocks, in general, will cause the observer to wait for longer before seeing the requisite number of blocks. To put this in our context, we observe that if $\beta_\ell(w) = s$ at time ℓ (so that $|w| = \ell$), then at least $2k - s$ block creation events must take place in order for the adversary to successfully create a double spend (which will expose $2k$ blocks to the observer). With this picture in place, we carry out the natural numerical evolution, conditioned on the value of β arising at $w = \ell$. (We specifically use $\beta_0 + a_0$.) This yields the results summarized in Figures 5 and 6, which are the analogues of Figures 3 and 4 for block-based settlement. A more detailed record of our results for block-based settlement is given in Table 2 in Appendix C.

Comparing settlement modes. In light of the comments above, it is particularly interesting to compare the Bitcoin settlement probability for the "wait for 6 blocks" rule against the (seemingly comparable) rule "wait for 60 minutes." We perform this comparison in Figure 7. More concretely, we consider the Bitcoin setting with 10 second delays and plot the upper bound on the temporal settlement

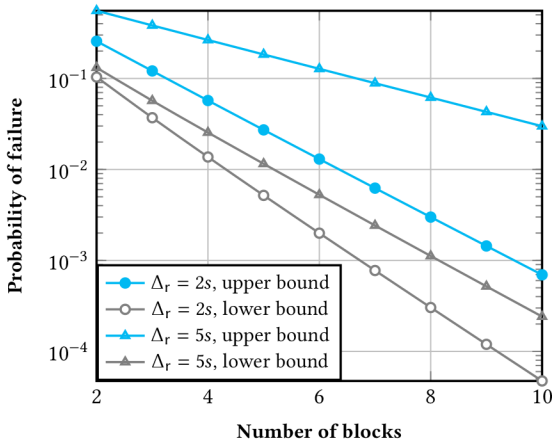

Figure 6: Ethereum block-based settlement failure for a 10% adversary.

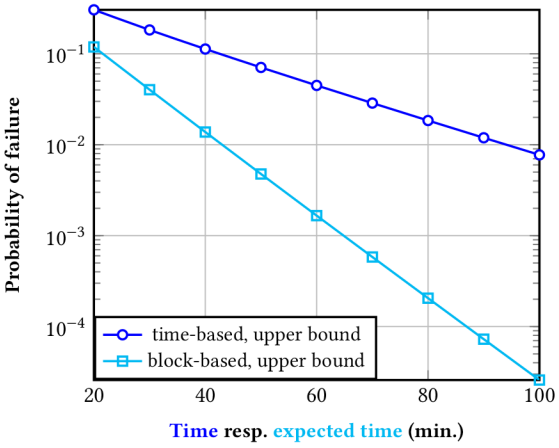

Figure 7: Bitcoin settlement method comparison: time- vs. block-based, for a 10% adversary and $\Delta_r = 10s$.

error as a function of time (as indicated by Figure 3), alongside with the upper bound on the block-based settlement error (as indicated by Figure 5) as a function of the expected time it takes for the particular number of blocks to appear under honest operation. As the graph illustrates, already in the above-mentioned case of 60 minutes vs. 6 blocks, the block-based settlement guarantees are an order of magnitude better. This illustrates that under normal operation of the protocol, users are able to arrive at their desired settlement guarantee significantly faster if they apply a block-based settlement rule.

REFERENCES

[1] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2020. Consensus Redux: Distributed Ledgers in the Face of Adversarial Supremacy. Cryptology ePrint Archive, Report 2020/1021. https://eprint.iacr.org/2020/1021.

[2] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2018. Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. In *ACM CCS 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, 913–930. https://doi.org/10.1145/3243734.3243848

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

[4] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Heidelberg, 66–98. https://doi.org/10.1007/978-3-319-78375-8_3

[5] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. 2020. Everything is a Race and Nakamoto Always Wins. In *ACM CCS 20*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 859–878. https://doi.org/10.1145/3372297.3417290

[6] Ethstats. 2021. https://ethstats.net/.

[7] Ittay Eyal and Emin Gün Sirer. 2014. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *FC 2014 (LNCS, Vol. 8437)*, Nicolas Christin and Reihaneh Safavi-Naini (Eds.). Springer, Heidelberg, 436–454. https://doi.org/10.1007/978-3-662-45472-5_28

[8] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *EUROCRYPT 2015, Part II (LNCS, Vol. 9057)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Heidelberg, 281–310. https://doi.org/10.1007/978-3-662-46803-6_10

[9] Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Tight Consistency Bounds for Bitcoin. In *ACM CCS 20*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 819–838. https://doi.org/10.1145/3372297.3423365

[10] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 3–16.

[11] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO 2017, Part I (LNCS, Vol. 10401)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, 357–388. https://doi.org/10.1007/978-3-319-63688-7_12

[12] Lucianna Kiffer, Rajmohan Rajaraman, and Shelat Abhi. 2018. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 729–744.

[13] Leslie Lamport. 2019. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*. 277–317.

[14] Jing Li and Dongning Guo. 2019. On Analysis of the Bitcoin and Prism Backbone Protocols in Synchronous Networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 17–24.

[15] Jing Li, Dongning Guo, and Ling Ren. 2020. Close Latency–Security Trade-off for the Nakamoto Consensus. *arXiv preprint arXiv:2011.14051* (2020).

[16] DSN Bitcoin Monitoring. 2021. https://dsn.tm.kit.edu/bitcoin/.

[17] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008).

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

[19] Ling Ren. 2019. Analysis of Nakamoto Consensus. Cryptology ePrint Archive, Report 2019/943. https://eprint.iacr.org/2019/943.

[20] Fred B Schneider. 1990. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)* 22, 4 (1990), 299–319.

[21] Yonatan Sompolinsky and Aviv Zohar. 2016. Bitcoin's security model revisited. *arXiv preprint arXiv:1605.09193* (2016).

[22] Jun Zhao, Jing Tang, Zengxiang Li, Huaxiong Wang, Kwok-Yan Lam, and Kaiping Xue. 2020. An analysis of blockchain consistency in asynchronous networks: Deriving a neat bound. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 179–189.

A THE SECURITY REGION

As already mentioned, we consider the Nakamoto PoW consensus in the by now standard setting, where honest and adversarial hashing successes appear according to (independent) Poisson processes with parameters r_h and r_a, respectively, and messages are delayed by at most Δ_r rounds. The security of the Nakamoto consensus in this model is determined by these three parameters (r_h, r_a, Δ_r), as previous works [5, 9] exactly identified the set of all parametrizations (r_h, r_a, Δ_r) that guarantee consistency except with negligible error, also called the *security region* of the protocol.

In this section, we explore the region of parameters for which the security of the protocol in the above asymptotic sense is guaranteed based on our method, we refer to this as the security region of our analysis. Towards that, we first present the following statement.

Lemma A.1. *Let $w = w_1, \ldots, w_n \in \{0, H\}^*$ and define $\mathcal{H}(w)$, the* height *of w, to be the quantity given by the recursive rule $\mathcal{H}(\epsilon) = 0$, $\mathcal{H}(H) = 1$,*

$$\mathcal{H}(0w) = \mathcal{H}(w)$$

and, for any symbol $x \in \{0, H\}$,

$$\mathcal{H}(Hxw) = 1 + \mathcal{H}(w) \ .$$

For $n > 0$, let $w_1, \ldots, w_n$ be independent random variables for which $\Pr[w_i = H] = p$ and $\Pr[w_i = 0] = (1 - p)$. Then

$$|\mathbb{E}[\mathcal{H}(w)] - \alpha n| \leq 1$$

for $\alpha = p/(1 + p)$.

Proof. Let $E_n = \mathbb{E}[\mathcal{H}(w_1, \ldots, w_n)]$. Then $E_n = p(E_{n-2} + 1) + (1-p)E_{n-1}$. The base cases, for $n = 0$ and $n = 1$, are straightforward computations. Then, by induction, we note that $|E_n - \alpha n|$ can be written

$$\begin{aligned}
&= |p(E_{n-2} + 1) + (1 - p)E_{n-1} - \alpha n| \\
&= |p(E_{n-2} + 1 - \alpha n) + (1 - p)(E_{n-1} - \alpha n)| \\
&\leq |p[E_{n-2} - \alpha(n-2)] + (1-p)[E_{n-1} - \alpha(n-1)] \\
&\quad + p - 2p\alpha - (1-p)\alpha \\
&\leq p|E_{n-2} - \alpha(n-2)| + (1-p)|E_{n-1} - \alpha(n-1)| \\
&\quad + |p - \alpha(1+p)| \\
&\leq p + (1-p) = 1 \,,
\end{aligned}$$

as desired. □

The security region of our analysis, as was the case with previous analysis [9], is determined by the behavior of β_0 when β_0 is sufficiently far from zero, in which case neutralization opportunities do not appear: in particular, so long as β_0 is negatively biased (and $(\beta_0, \beta_{\mathsf{H}}, \beta_{\mathsf{h}})$ cross zero with constant probability in situation where $\beta_0 \approx 0$), the analysis provides consistency except with exponentially decaying error probability. To understand the security region of our analysis, we hence simply need to identify the bias of the behavior of β_0 when β_0 is sufficiently far from zero, and require that this bias be negative.

Over any time interval T, the number of adversarial successes is given by the Poisson distribution with parameter Tr_a and has expectation exactly Tr_a. With Δ_{r} network delay, the probability of at least one honest success in a slot of length Δ_{r} is $p \triangleq 1 - \exp(-\Delta_{\mathrm{r}} r_h)$; it follows from Lemma A.1 that the expected height $\mathcal{H}$ of the characteristic vector of honest successes in time T (resulting in T/Δ_{r} slots), is

$$\frac{T}{\Delta_{\mathrm{r}}} \cdot \frac{p}{1+p} = \frac{T}{\Delta_{\mathrm{r}}} \cdot \frac{1 - \exp(-\Delta_{\mathrm{r}} r_h)}{2 - \exp(-\Delta_{\mathrm{r}} r_h)} \,,$$

thus the analysis provides security if

$$r_a < \frac{1 - \exp(-\Delta_{\mathrm{r}} r_h)}{\Delta_{\mathrm{r}}(2 - \exp(-\Delta_{\mathrm{r}} r_h))} \ .$$

This, as expected, falls short of the optimal, tight security region $r_a < 1/(\Delta_{\mathrm{r}} + 1/r_h)$ established in two recent articles [5, 9]. Nonetheless, it is still better than the security region $r_a < r_h \exp(-2\Delta_{\mathrm{r}} r_h)$ of arguments based on leveraging Δ_{r}-isolated blocks, present in earlier analyses [15, 19]. We reemphasize that our objective in this work was not to match the asymptotically tight security region (as that was already achieved in prior work), but rather to devise an analysis that allows for concrete, practically relevant settlement bounds.

B OMITTED PROOFS

B.1 Proof of Lemma 3.2

Proof. We proceed by induction on the length of w. The base case is immediate, as the trivial fork has no nontrivial tines.

We begin by establishing the lower bounds for the quantities $\beta_\ell(ws)$ for $s \in \Sigma_{\mathsf{mh}}$ corresponding to each of the equations (2) above. These implicitly yield an optimal (on-line) adversary for maximizing $\beta_\ell()$: specifically, for a characteristic string $w_1, \ldots, w_n$, this yields a sequence of forks $F_1 \sqsubseteq \cdots \sqsubseteq F_n$ so that $F_t \vdash w_1 \ldots w_t$ and each F_t is only determined by the string $w_1 \ldots w_t$. We then turn to the corresponding upper bounds, which establish equality in each of the cases above.

Bounding from below; the optimal adversary. Let w be a characteristic string and $F \vdash w$ a fork achieving $\beta_\ell(F) = \beta_\ell(w)$; let $T \nsim_\ell T^*$ be two tines of F which witness β so that T^* is dominant and $\alpha_F(T) = \beta_\ell(F)$. Then:

(1) Consider the fork $F' \vdash w\mathsf{a}$ obtained by extending the tine T with a new adversarial vertex labeled with the last symbol. This new fork achieves $\beta_\ell(F') \geq \beta_\ell(w) + 1$.
(2) Consider the fork $F' \vdash w\mathsf{h}$ obtained by adding an honest vertex to the end of T^*. This new fork has $\mathsf{len}(\overline{F'}) = \mathsf{len}(\overline{F}) + 1$ and achieves $\beta_\ell(F') \geq \beta_\ell(w) - 1$. (A pair of tines that witness this in F' are T and the dominant tine terminating in the new vertex.) An analogous argument also works when $s = \mathsf{H}$, the only difference is that T^* is extended by two honest vertices of the same depth.
(3) If $s = \mathsf{h}$ and $|w\mathsf{h}| < \ell$, consider an arbitrary fork $F' \vdash w\mathsf{h}$ for which $F \sqsubseteq F'$. As $|w\mathsf{h}| < \ell$ any dominant tine in F' can serve as both T^* and T in the definition of β_ℓ. This achieves $\beta_\ell(F') \geq 0$, and hence, if $\beta_\ell(w) = 0$ then $\beta_\ell(w\mathsf{h}) \geq \beta_\ell(w)$ as desired.
(4) If $s = \mathsf{H}$ and $\beta_\ell(w) = 0$, consider the fork $F' \vdash w\mathsf{H}$ obtained by adding one honest vertex to the end of each of the tines T and T^* (note that $\mathsf{len}(T) = \mathsf{len}(T^*)$). This new fork has $\mathsf{len}(\overline{F'}) = \mathsf{len}(\overline{F}) + 1$ and $\beta_\ell(F') \geq 0$, as witnessed by the two tines terminating in the newly added vertices.

Bounding from above. To complete the proof, we establish the opposite inequalities in each of the cases $s \in \{\mathsf{h}, \mathsf{H}, \mathsf{a}\}$.

The case $s = \mathsf{h}$. Let $F' \vdash w\mathsf{h}$ be a fork for which $\beta_\ell(F') = \beta_\ell(w\mathsf{h})$; let T and T^* be tines that witness this value of β, where T^* is dominant and $\alpha_{F'}(T) = \beta_\ell(F')$. Let $F \vdash w$ be the fork obtained by removing the honest vertex v of F' associated with the final h symbol. Observe that $\mathsf{len}(\overline{F}) \leq \mathsf{len}(\overline{F'}) - 1$.

If v does not appear on T, this tine T remains in F and $\alpha_F(T) \geq \beta_\ell(w\mathsf{h}) + 1$. Note the tine T^* might not appear in F if v appears on

T^*. In any case, however, the restriction of T^* to the fork F always has length at least $\mathsf{len}(\overline{F})$ and hence is dominant. We conclude that $\beta_\ell(w) \geq \beta_\ell(w\mathsf{h}) + 1$.

Otherwise v appears on T, in which case T is an honest tine and $\beta_\ell(w\mathsf{h}) = \alpha_{F'}(T) = 0$. If the tines T and T^* are distinct, we may switch their roles (as both are dominant) and apply the argument above to conclude that $\beta_\ell(w) \geq \beta(w\mathsf{h}) + 1$. Otherwise $T = T^*$ and we conclude that $|w\mathsf{h}| < \ell$. In this case, removing the last vertex from these tines results in a tine $\check{T}$ for which $\alpha_F(\check{T}) = 0$, establishing $\beta_\ell(w) \geq 0$. Hence, considering separately the cases $\beta_\ell(w) > 0$ and $\beta_\ell(w) = 0$, in each of them the desired inequality holds.

The case $s = \mathsf{H}$**.** The proof is very similar to the previous case: given a fork $F' \vdash w\mathsf{H}$ and its witness tines T and T^*, a new fork $F \vdash w$ is constructed by removing all honest vertices in F' associated with the final H symbol (denote these $\mathcal{V}$). We know that

$$\mathsf{len}(\overline{F}) \leq \mathsf{len}(\overline{F'}) - d \,, \tag{9}$$

where d is the maximum number of vertices from $\mathcal{V}$ that appear on the same tine in F' (note that contrary to the case $s = \mathsf{h}$, we can have $d > 1$).

If either T or T^* contains no vertices from $\mathcal{V}$ then $\beta_\ell(w) \geq \beta_\ell(w\mathsf{H}) + 1$ can be established by an argument identical to the previous case.

On the other hand, if both T and T^* contain some vertices from $\mathcal{V}$, the argument is similar to the case $T = T^*$ above. Namely, we know that $\beta_\ell(w\mathsf{H}) = 0$ (as T is honest), and let $\tilde{T}$ denote the tine out of T and T^* that contains at least as many vertices from $\mathcal{V}$ as the other one (in case of equality, choose arbitrarily). Then again, after removing all vertices $\mathcal{V}$ from F' to obtain F, the tine that remained from $\tilde{T}$ still has length at least $\mathsf{len}(\overline{F})$ thanks to (9), and so the remainders of the two tines T and T^* in F witness $\beta_\ell(w) \geq \beta_\ell(F) \geq 0$ as desired.

The case $s = \mathsf{a}$**.** Let $F' \vdash w\mathsf{a}$ realize $\beta_\ell(F') = \beta_\ell(w\mathsf{a})$ and let T and T^* be two tines of F' that witness $\beta_\ell(F')$, as above.

We begin with an argument showing that the fork F' can be restructured to yield a fork $\hat{F} \vdash w\mathsf{a}$ for which $\beta_\ell(\hat{F}) = \beta_\ell(F')$ and there is a pair of tines $\hat{T}$ and $\hat{T}^*$ witnessing this value of $\beta_\ell(\hat{F})$ with the property that $\hat{T}$ terminates with the adversarial vertex v associated with the last symbol a, $\alpha_{\hat{F}}(\hat{T}) = \beta_\ell(w\mathsf{a})$, and $\hat{T}^*$ is dominant.

Restructuring F'. If T contains the vertex v, F' already has the desired property. Otherwise the vertex v must, in fact, appear on T^*: if it appeared on neither T nor T^*, removing the vertex from the end of the tine on which it appears (if it exists) and adding it to the end of the tine T would result in a larger $\beta_\ell()$. To construct the fork $\hat{F}$, let v_h denote the honest vertex of maximum depth among those vertices on either T or T^*. Let $T_h \in \{T, T^*\}$ be a tine containing v_h, and T_a denote the other tine. $\hat{F}$ is constructed from F' as follows: (I.) All adversarial vertices on T_h appearing after v_h are removed from T_h and inserted into the tine T_a, producing a new tine $\hat{T}$; this is possible because T_a has no honest vertex with label larger than $\mathsf{l}_\#(v_h)$. Observe that the tine $\hat{T}$ constructed in this way contains the final adversarial vertex v. (II.) Starting from the vertex v_h, construct a tine $\hat{T}^*$ by going over the sequence of slots $i \in \{\mathsf{l}_\#(v_h), \ldots, |w|\}$ and (i) if $w_i = \mathsf{h}$, move the unique honest vertex labeled i in F' on top of the constructed tine; and (ii) if $w_i = \mathsf{H}$, remove all vertices labeled i in F' and instead, add two vertices of equal depth on top of the growing tine (to satisfy axiom (A3)). Out of these two tips, choose arbitrarily to continue the iterative process. As a special case, in the initial step $i = \mathsf{l}_\#(v_h)$, if $w_i = \mathsf{h}$ then no modification is done; if $w_i = \mathsf{H}$ then all honest children of v_h with label i are removed (and an equal-depth honest sibling to v_h is added). Any (necessarily adversarial) vertices orphaned by this process can be attached to the fork arbitrarily. This constructs a new fork $\hat{F} \vdash w\mathsf{a}$.

As $T \nearrow_\ell T^*$, it is clear that the tines $\hat{T}$ and $\hat{T}^*$ constructed above inherit this property. Note, also, that $\hat{T}^*$ is clearly dominant in $\hat{F}$, as it terminates with the deepest honest vertex of $\hat{F}$. It remains to ensure that $\alpha_{\hat{F}}(\hat{T}) \geq \beta_\ell(F')$. Recall that $T_a \in \{T, T^*\}$. If $T_a = T$, it is clear that $\alpha_{\hat{F}}(\hat{T}) \geq \alpha_{F'}(T)$ because $\mathsf{len}(\overline{\hat{F}}) \leq \mathsf{len}(\overline{F'})$ and $\mathsf{len}(\hat{T}) \geq \mathsf{len}(T)$ by construction. In the other case $T_a = T^*$, any adversarial vertices were inserted into the tine T^* (to yield $\hat{T}$). Recall that T^* was dominant in F'; if $\beta_\ell(w\mathsf{a}) \leq 0$, this immediately yields $\alpha_{\hat{F}}(T^*) \geq \beta_\ell(w\mathsf{a})$, as desired. Otherwise, observe that the number of adversarial vertices inserted in T^* is at least $\alpha_{F'}(T) = \beta_\ell(w\mathsf{a})$, in which case it is clear that $\alpha_{\hat{F}}(\hat{T}) \geq \beta_\ell(F')$, as desired. This completes the construction and its analysis.

To complete the argument, assume that the fork F' possesses the property guaranteed above (that the final adversarial vertex appears on the tine T). Let $F \sqsubseteq F'$ denote the fork $F \vdash w$ obtained by removing the adversarial vertex v associated with the final symbol a. Then the restriction of T to F and the tine T^* together witness the fact that $\beta_\ell(w) \geq \beta_\ell(F) \geq \beta_\ell(F') - 1 = \beta_\ell(w\mathsf{a}) - 1$, as desired. □

B.2 Proof of Lemma 4.3

The following auxiliary statement is a general fact about witness forks and is not specific to our investigation, it will however prove useful in establishing Lemma 4.3.

Lemma B.1. *Let $w \in \Sigma_\infty^n$. Then there exists a fork F^* that is a witness 1-fork for w (i.e., $F^* \vdash_1 w$ and $\beta_\ell^1(F^*) = \beta_\ell^1(w)$) that satisfies*

$$\mathsf{len}(\overline{F^*}) \leq \mathsf{len}(\overline{F^*_{\lceil 1}}) + 1 \,. \tag{10}$$

Proof. Let $F \vdash_1 w$ be any witness fork, we show how it can be transformed into F^*. The fork $F^* \vdash_1 w$ is constructed as follows. For any $d \in \mathbb{N}$, let V_d denote the set of all vertices in F with depth d. First, we partition the set of depths $D \triangleq \{\mathsf{len}(\overline{F_{\lceil 1}}) + 1, \ldots, \mathsf{len}(\overline{F})\} \subseteq \mathbb{N}$ into two sets D_a and D_h, where D_a contains all depths in which F only has adversarial vertices, while D_h are the depths in which there are also some honest vertices: formally $D_\mathsf{h} \triangleq \{d \in D \mid \exists v \in V_d \colon \mathsf{l}_{\mathsf{type}}(v) = \mathsf{h}\}$ and $D_\mathsf{a} \triangleq D \setminus D_\mathsf{h}$. If $D_\mathsf{h} = \emptyset$ then we can simply leave $F^* = F$, as F has the desired property (10). Otherwise, let D'_h be D_h with its minimum element removed. We obtain F^* from F via two modifications:

(i) Remove from F all vertices with depths in $D_\mathsf{a} \cup D'_\mathsf{h}$, along with their associated edges. For each vertex v that was a child of some removed vertex, add an edge to make it a child of its deepest ancestor in F that was not removed.

Let v_h be some honest vertex in F with $\mathsf{len}(v_\mathsf{h}) = \min D_\mathsf{h}$ (note that several such vertices may exist). By definition of D'_h, $\mathsf{len}(v_\mathsf{h}) \notin D'_\mathsf{h}$ and v_h was not removed in the first step.

(ii) If the last symbol of w is $w_n = (\mathsf{H}, a)$ for some $a \in \mathbb{N}$, i.e., if w asks for at least 2 honest vertices with the $\mathsf{l}_\#$-label equal to n, add a new honest vertex v'_h to F^*, with $\mathsf{l}_\#(v'_\mathsf{h}) = n$, $\mathsf{l}_{\mathsf{type}}(v'_\mathsf{h}) = \mathsf{h}$, and v'_h having the same parent as v_h has after modification (i).

We first need to verify that $F^* \vdash_1 w$ and that it satisfies (10). It is straightforward to verify that modifications (i) and (ii) maintain axioms (A1)–(A3); in particular, modification (ii) guarantees that F^* contains the correct number of honest vertices with $\mathsf{l}_\#(v) = n$ as required by axiom (A3). Finally, by construction of modification (i), we know that the only honest vertices in F^* with depth greater than $\mathsf{len}(\overline{F_{\lceil 1}})$ are those that originate from honest vertices in F with depth in $D_\mathsf{h} \setminus D'_\mathsf{h}$ (or the newly added vertex v'_h), and the depth of all these vertices in F^* must be $\mathsf{len}(\overline{F_{\lceil 1}}) + 1$; this implies both axiom (A4) and property (10).

Finally, it remains to argue that F^* is a witness fork for w. To see this, consider any pair (T^*, T) of witness tines in F: clearly, as a result of the above two modifications, $\mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F}) - |D_\mathsf{a} \cup D'_\mathsf{h}|$, while $\mathsf{len}(T)$ decreased by at most $|D_\mathsf{a} \cup D'_\mathsf{h}|$ depending on whether it contained vertices of all depths in $D_\mathsf{a} \cup D'_\mathsf{h}$. At the same time, the relation $T \nsim_\ell T^*$ was not violated by the modifications as the only effect they had on T and T^* was removing vertices. This proves that $\beta^1_\ell(F^*) \geq \beta^1_\ell(F)$, implying $\beta^1_\ell(F^*) = \beta^1_\ell(w)$ as desired. □

We can now proceed to prove Lemma 4.3.

Proof. Let F^* be a witness 1-fork for w satisfying (10), as guaranteed by Lemma B.1. We construct another fork $F^+ \vdash_1 w$ from it that will be useful in our proof. Let (T^*, T) be a pair of witness tines in F^*, i.e., $\mathsf{len}(T^*) = \mathsf{len}(\overline{F_{\lceil 1}})$, $T^* \nsim_\ell T$, and $\beta^1_\ell(F^*) = \alpha^1_{F^*}(T) = \mathsf{len}(T) - \mathsf{len}(\overline{F^*_{\lceil 1}})$. If $\mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F^*_{\lceil 1}})$ simply let $F^+ := F^*$, otherwise we have $\mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F^*_{\lceil 1}}) + 1$. This means the last honest slot n has at least one honest success. In this case, let T^*_n be some longest honest tine in F^*; we have that $\mathsf{len}(T^*_n) = \mathsf{len}(\overline{F^*}) = \mathsf{len}(T^*) + 1$, and the terminating vertex of T^*_n (call it v_n) is honest and has $\mathsf{l}_\#(v_n) = n$. We consider two subcases: if $T^*_n \nsim_\ell T$ then we let $F^+ := F^*$; otherwise we construct F^+ from F^* as follows: the vertex v_n is moved so that it extends the tine T^* (this operation does not change the depth of v_n), and all direct children of v_n become direct children of the parent of v_n (this decreases by one the depth of all descendants of v_n, which are all adversarial). This concludes the definition of F^+; observe that $F^+ \vdash_1 w$ as the validity of all fork axioms is inherited from F^*. Moreover, we have $\beta^1_\ell(F^+) \geq \beta^1_\ell(F^*) - 1$ as the depth of some adversarial vertices possibly decreased by 1 and these vertices don't share the same branch with T^*; hence, $\mathsf{len}(T^*)$ remained unaffected and $\mathsf{len}(T)$ decreased by at most 1 from F^* to F^+.

Let $w' \in \mathcal{D}_1(w)$ and $F' \equiv F^+$ be such that $F' \vdash_0 w'$ as guaranteed by Lemma 4.2. Then we have

$$\beta^1_\ell(w) = \beta^1_\ell(F^*) \leq \beta^1_\ell(F^+) + 1 \overset{(a)}{\leq} \beta^0_\ell(F') + 2 \leq \beta^0_\ell(w') + 2 \leq \max_{w' \in \mathcal{D}_1(w)} \beta^0_\ell(w') + 2$$

as desired. Inequality (a) needs some justification. We will explain now that it follows from the construction of F^+. Indeed, if $\mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F^*_{\lceil 1}})$ and hence $F^+ = F^*$, we have $\beta^0_\ell(F') \geq \alpha^0_{F'}(T) = \alpha^1_{F^+}(T) = \beta^1_\ell(F^+)$ as desired. Similarly, if $\mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F^*_{\lceil 1}}) + 1$ and $T^*_n \nsim T$, then the pair (T^*, T) is witnessing $\beta^0_\ell(F') \geq \alpha^0_{F'}(T) = \alpha^1_{F^+}(T) + 1 = \beta^1_\ell(F^+) + 1$, as $\mathsf{len}(\overline{F'}) = \mathsf{len}(\overline{F^*}) = \mathsf{len}(\overline{F^*_{\lceil 1}}) + 1 = \mathsf{len}(\overline{F^+_{\lceil 1}}) + 1$. Finally, if $T^*_n \sim T$ in F^*, then the surgery performed on the terminating vertex v_n of T^*_n ensures that after the modification (i.e., in F^+), the tine ending in v_n (call it T^*_n) satisfies $T^*_n \nsim_\ell T$ in F^+. Hence, $T^*_n \nsim_{\ell+1} T$ in F' and therefore can witness $\beta^0_{\ell+1}(F') \geq \alpha^0_{F'}(T) = \alpha^1_{F^+}(T) + 2 = \beta^1_\ell(F^+) + 2$, as then we again have $\mathsf{len}(\overline{F'}) = \mathsf{len}(\overline{F^+_{\lceil 1}}) + 1$, but also $\mathsf{len}_{F'}(T) = \mathsf{len}_{F^+}(T) \geq \mathsf{len}_{F^*}(T) - 1$. This justifies inequality (a) in all three cases of the construction of F^+ and concludes the proof. □

B.3 Proof of Lemma 4.4

Proof. Consider $(\beta, a), (\bar\beta, \bar a) \in \mathbb{Z} \times \mathbb{N}$ such that $(\beta, a) \preceq (\bar\beta, \bar a)$. The case $(\beta, a) = (\bar\beta, \bar a)$ is immediate, hence it remains to consider either (i) $\beta + a < \bar\beta + \bar a$, or (ii) $\beta + a = \bar\beta + \bar a \wedge a < \bar a$. Whenever a function $f\colon \mathbb{Z} \times \mathbb{N} \to \mathbb{Z} \times \mathbb{N}$ will be clear from the context, we will denote by a star the components of the image under this function, e.g., $(\beta^*, a^*) := f(\beta, a)$ and $(\bar\beta^*, \bar a^*) := f(\bar\beta, \bar a)$. We need to show that

$$f(\beta, a) = (\beta^*, a^*) \preceq (\bar\beta^*, \bar a^*) = f(\bar\beta, \bar a) \tag{11}$$

for all three functions f from the statement.

For the function NHE, in case (i) we clearly have

$$\mathsf{NHE}(\beta, a, a') = (\beta + a, a') \prec (\bar\beta + \bar a, a') = \mathsf{NHE}(\bar\beta, \bar a, a')$$

as $\beta + a + a' < \bar\beta + \bar a + a'$ in this case. In case (ii), we have $\mathsf{NHE}(\beta, a, a') = \mathsf{NHE}(\bar\beta, \bar a, a')$, as desired. The proof is identical for the function HE.

Consider now the function NO that is defined in terms of two underlying functions $f_1(\beta, a) \triangleq (\max\{0, \beta + a\}, a' + \min\{0, \beta + a\})$ and $f_2(\beta, a) \triangleq \mathsf{HE}(\beta, a, a') \triangleq (\beta + a - 1.a')$ and a predicate $\mathsf{p}(\beta, a)$ indicating whether or not $\beta \in \{-a - a', \ldots, 0\}$. We first argue that both f_1 and f_2 are monotone. The monotonicity of f_2 has already been established as f_2 is simply HE. For f_1, we have

$$\begin{aligned} \beta^* + a^* &= \beta + a + a' \\ a^* &= a' + \min\{0, \beta + a\} \end{aligned}$$

and hence in case (i) we have $\beta^* + a^* < \bar\beta^* + \bar a^*$, while in case (ii) we have $\beta^* + a^* = \bar\beta^* + \bar a^*$ and $a^* = \bar a^*$. In both cases we have $f_1(\beta, a) \preceq f_1(\bar\beta, \bar a)$ as desired. This proves (11) whenever $\mathsf{p}(\beta, a) = \mathsf{p}(\bar\beta, \bar a)$.

It remains to argue that (11) also holds for NO if $\mathsf{p}(\beta, a) \neq \mathsf{p}(\bar\beta, \bar a)$, we show this via case analysis. Consider the two possible cases:

Case $\neg\mathsf{p}(\beta, a) \wedge \mathsf{p}(\bar\beta, \bar a)$: We have $\beta^* + a^* = \beta + a + a' - 1 < \bar\beta + \bar a + a' = \bar\beta^* + \bar a^*$, implying (11) as desired.

Case $\mathsf{p}(\beta, a) \wedge \neg\mathsf{p}(\bar\beta, \bar a)$: In this case we cannot have (ii), as the condition (ii) and $\mathsf{p}(\beta, a)$ together imply $\mathsf{p}(\bar\beta, \bar a)$. Therefore, we have (i) and $\beta^* + a^* = \beta + a + a' \leq \bar\beta + \bar a + a' - 1 = \bar\beta^* + \bar a^*$ and at the same time $a^* = a' + \min\{0, \beta + a\} \leq a' = \bar a^*$, again implying (11).

This concludes the proof also for NO.

Finally, observe that HE^t_ℓ is either HE or NO depending on the (fixed) parameters t and ℓ, and hence the above also implies the monotonicity of HE^t_ℓ and concludes the proof. □

Time (min)	upper bounds		lower bounds	
	$\Delta_r = 10s$	$\Delta_r = 4s$	$\Delta_r = 10s$	$\Delta_r = 4s$
	Bitcoin; 10% adversary			
20	0.304519	0.298281	0.295002289	0.294517617
30	0.182942	0.177858	0.175313516	0.175442853
40	0.112861	0.109011	0.10716052	0.106774039
50	0.0707863	0.0679664	0.066656956	0.066352072
60	0.0448913	0.0428636	0.041949497	0.041845542
70	0.0286956	0.0272542	0.026621323	0.026448959
80	0.0184528	0.0174364	0.017000505	0.016874481
90	0.011922	0.0112094	0.010910293	0.010819242
100	0.00773178	0.00723447	0.007029835	0.006964646
	Bitcoin; 20% adversary			
20	0.505249	0.498112	0.494926979	0.492990797
30	0.383382	0.376117	0.373064366	0.371644562
40	0.295733	0.288859	0.286104409	0.284040603
50	0.230435	0.224161	0.221742068	0.219810527
60	0.180805	0.175197	0.173106136	0.17163273
70	0.142594	0.137653	0.135862886	0.134299714
80	0.11291	0.1086	0.107077575	0.105704297
90	0.0896948	0.0859628	0.084675044	0.083480911
100	0.0714434	0.0682312	0.067145976	0.066115746

Confs.	upper bounds		lower bounds	
	$\Delta_r = 10s$	$\Delta_r = 4s$	$\Delta_r = 10s$	$\Delta_r = 4s$
	Bitcoin; 10% adversary			
2	0.118882	0.111154	0.091072133	0.090289244
3	0.0402842	0.0368385	0.029544274	0.029154201
4	0.0137891	0.0123524	0.009793747	0.009616722
5	0.00476516	0.00418514	0.003294434	0.003217994
6	0.00165992	0.00143009	0.001120043	0.00108804
7	0.000582003	0.000492027	0.000383901	0.000370782
8	0.000205151	0.000170222	0.000132434	0.000127138
9	7.2633e-05	5.91573e-05	4.5926E-05	4.38129E-05
10	2.58108e-05	2.06366e-05	1.5996E-05	1.51607E-05
	Bitcoin; 20% adversary			
2	0.466437	0.45271	0.319646859	0.317452323
3	0.288865	0.277594	0.188866365	0.186940123
4	0.177784	0.169269	0.113180954	0.111647534
5	0.109524	0.103348	0.068498618	0.06733905
6	0.0676876	0.0633137	0.041762682	0.040912987
7	0.0419841	0.0389337	0.025608615	0.024999029
8	0.0261309	0.0240265	0.015775657	0.015344919
9	0.016314	0.0148737	0.009755295	0.009454384
10	0.0102126	0.00923299	0.006051757	0.005843401

Time (min)	upper bounds		lower bounds	
	$\Delta_r = 5s$	$\Delta_r = 2s$	$\Delta_r = 5s$	$\Delta_r = 2s$
	Ethereum; 10% adversary			
2	0.137626	0.0279521	0.015828578	0.011812983
3	0.0527935	0.00548293	0.002145191	0.001584263
4	0.0203159	0.0010971	0.000410932	0.000251815
5	0.00782799	0.000221883	6.9340E-05	3.615E-05
6	0.003018	4.51668e-05	1.0273E-05	5.61563E-06
7	0.00116389	9.23251e-06	2.0634E-06	9.38321E-07
8	0.00044892	1.89193e-06	3.6112E-07	1.48653E-07
9	0.000173164	3.87677e-07	5.5071E-08	2.23033E-08
10	6.67978e-05	7.87459e-08	1.1272E-08	3.57683E-09
	Ethereum; 20% adversary			
2	0.384056	0.156394	0.117232788	0.092808469
3	0.245871	0.0697603	0.043623877	0.033041265
4	0.158287	0.0317031	0.019425505	0.012949768
5	0.102233	0.0145709	0.008178393	0.004849916
6	0.0661652	0.00674751	0.003249166	0.001899153
7	0.0428818	0.00314153	0.001500296	0.000774062
8	0.0278188	0.00146854	0.000650086	0.00030802
9	0.0180596	0.000688627	0.000264363	0.000119565
10	0.0117302	0.000323706	0.000123975	4.80722E-05

Table 1: The failure probability of the temporal settlement rule for Bitcoin (top) and Ethereum (bottom) under different settlement time, adversary ratio and network delays.

Confs.	upper bounds		lower bounds	
	$\Delta_r = 5s$	$\Delta_r = 2s$	$\Delta_r = 5s$	$\Delta_r = 2s$
	Ethereum; 10% adversary			
2	0.554298	0.256406	0.13124146	0.103613076
3	0.38244	0.120911	0.056912431	0.037008885
4	0.264554	0.0571909	0.025438212	0.013695484
5	0.183481	0.0271947	0.011522827	0.005185141
6	0.12746	0.0129908	0.005263762	0.001993565
7	0.0886243	0.00622754	0.002419679	0.000774795
8	0.0616519	0.00299308	0.001117841	0.000303492
9	0.0428996	0.00144124	0.000518528	0.000119582
10	0.0298552	0.000694936	0.000241348	4.73342E-05
	Ethereum; 20% adversary			
2	1.03875	0.673397	0.410871826	0.351521687
3	0.889277	0.479654	0.282740403	0.219222634
4	0.749407	0.337735	0.198320599	0.139166224
5	0.628655	0.237452	0.140561678	0.089448203
6	0.526782	0.167208	0.100281703	0.058018531
7	0.441386	0.118012	0.071879557	0.037893876
8	0.369901	0.0834758	0.051707349	0.024883864
9	0.310061	0.0591607	0.037305049	0.016411126
10	0.259949	0.0419967	0.026980289	0.010861279

Table 2: The failure probability of the block-based settlement rule for Bitcoin (top) and Ethereum (bottom) under different number of confirmations, adversary ratio and network delays.

C DETAILED EXPLICIT BOUNDS

In Table 1, we provide a detailed account of our numerical estimates of the failure probability of time-based settlement in both main blockchains of interest, Bitcoin and Ethereum. We give both lower and upper bounds provided by our method, and consider two variants of adversarial power (10% and 20%) and two variants of the bound on network delay ($\Delta_r \in \{4\text{ sec}, 10\text{ sec}\}$ for Bitcoin and $\Delta_r \in \{2\text{ sec}, 5\text{ sec}\}$ for Ethereum). In Table 2, we provide similar estimates of the failure probabilities for block-based settlement.