A Better Proof-of-Work Fork Choice Rule

Dionysis Zindros[1,2], Apostolos Tzinas[2,3],
Karl Kreder[4], Shreekara Shastry[4], and Sriram Vishwanath[4,5]

[1] Stanford University
[2] Common Prefix
[3] National Technical University of Athens
[4] Dominant Strategies
[5] University of Texas at Austin

Abstract. We propose a modification to the fork choice rule of proof-of-work blockchains. Instead of choosing the heaviest chain, we choose the chain with the most *intrinsic work*. The intrinsic work of a block is roughly the number of zeroes at the front of its hash. This modification allows us to safely speed up the protocol, yielding a roughly 40% improvement in *confirmation delay* as compared to Bitcoin for adversaries close to 10%. Our modification is at the level of the proof-of-work inequality, and thus can be composed with any other methods to improve latency proposed in the literature (e.g., GHOST). We compile detailed simulation evidence from 3,000 years of simulated executions of our system across different parameters. We formally prove the security of our new protocol in the Bitcoin Backbone model. These proofs use a new technical tool, the *real-valued Random Oracle* which may be of independent interest.

1 Introduction

In Bitcoin [24], a valid block must satisfy the *proof-of-work* inequality $H(B) < T$, where H is a hash function and T is a small *target*. Simply put, valid blocks must have hashes that begin with a desired number of 0s. Some blocks satisfy the inequality better than others: They have a bunch of *extra* zeroes at the front of their hash. Nevertheless, these "heavier" blocks are counted all the same when choosing which chain to mine on. We posit that the weight of a block is information that can be useful to improve the protocol. In this paper, we introduce *PoEM*. We modify the fork choice rule of Bitcoin to take this information into account, achieving better confirmation latency.

Construction overview. Miners still mine on the heaviest chain. We only change how chains are scored. In Bitcoin, every block counts for the same work[6]. In PoEM, we give each block a score equal to the number of *extra* zeroes at the front of its hash, a value we call its *intrinstic work*. Honest parties adopt the chain with the most total intrinsic work.

[6] Bitcoin blocks can count differently when difficulty adjusts, but count the same during the same epoch. Our analysis is in the static population setting [10], where the population does not change with time.

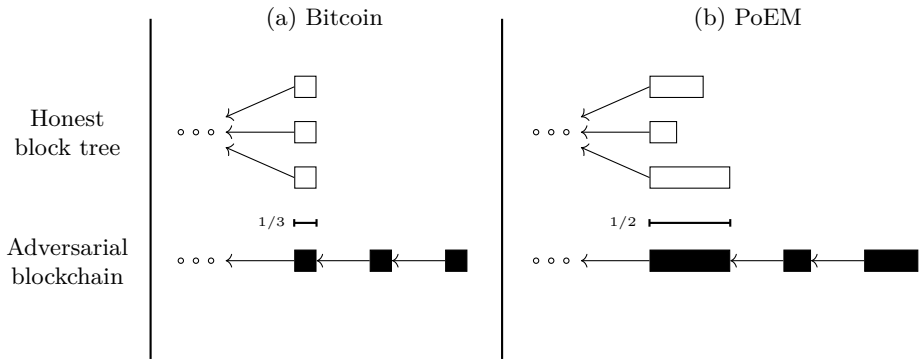


Fig. 1: A Bitcoin (left) and a PoEM (right) execution where the honest parties (top) and the adversary (bottom) are handed the same blocks. The honest PoEM tree grows faster than the honest Bitcoin tree as compared to the adversarial chain.

To guarantee security, it suffices [5] that the work of any honest chain grows faster than the work of the adversary's chain. Suppose we have n blocks produced in quick succession. If they are honestly produced, they will be placed in parallel due to network delays, forming n forks. If they are adversarially produced, since no network delay is incurred, they will be placed in series, forming a chain of length n. In Bitcoin, all blocks count the same constant amount of work w. So the honest chain grows by w, and the adversarial chain grows by nw. In PoEM, each block is scored differently: $w_1, w_2, \ldots, w_n$, but has an expected work of w. The adversarial chain grows by the sum of its blocks' work, $\sum_{i=1}^{n} w_i$. In expectation, this is equal to nw, same as Bitcoin. However, the honest chain now grows by the heaviest block's work: $\max_i w_i$, which in expectation is greater than w. Hence, in expectation, honest PoEM chains grow faster than honest Bitcoin chains in relation to the adversary's chain.

Let's look at the example in Figure 1. The same blocks, with the same amount of work, are given to both the adversary and the honest parties. The adversary places all her blocks in series, while the honest blocks are necessarily placed in parallel due to network delay. In Bitcoin, the adversary makes 3 times the progress that the honest parties make. However, in PoEM, the adversary only makes 2 times the progress that the honest parties make. This means that honest work is more effectively utilized in PoEM than in Bitcoin. Less honest work is wasted.

Our contributions.

- We construct a protocol which retains the same level of security as Bitcoin, while achieving better confirmation latency because the block production rate can be safely increased.
- We simulate over 3,000 years of continuous-time executions of both Bitcoin and PoEM using a combination of numeric and analytic techniques across a

range of parameters and conclude that PoEM is 40% faster in latency than Bitcoin for a 10% adversary.
- We prove the security of PoEM in the Bitcoin Backbone [10] model. This requires novel techniques such as the *real-valued random oracle*, a hash model that returns a *continuous* value. We prove it behaves identically to the usual random oracle with overwhelming probability.
- We deployed a production testnet in which more than 2,000 miners from the community participated with an average hash rate of 50 GH/s using ProgPoW [23].

Related work. Bitcoin was first proven secure in the static population setting [10], and later also studied in the variable population setting [11]. The idea of using a more nuanced proof-of-work inequality in which some blocks are considered heavier than others was first put forth by Andrew Miller [22], with the first complete protocol to utilize it being Proofs of Proof-of-Work [16]. These were later refined multiple times to account for non-interactivity [18], backwards compatibility [19], onlineness [17], on-chain data efficiency [15], gas consumption [4], bribing resilience [31], and variable populations [30]. We are the first to modify the fork choice rule to take these refinements into account, following our previous short paper "POEM: Proof of Entropy Minima" [21], where the entropic fork choice rule was defined but not analyzed. Our work only changes the PoW inequality. Other mechanisms refining the fork choice rule are orthogonal and can be combined with our approach, yielding even further performance gains. Such examples include GHOST [28], PHANTOM [29], SPECTRE [26], and GhostDAG [27]. Additional mechanisms towards improving the latency and throughput of proof-of-work blockchains at the consensus layer, also composable with ours, include parallel chains [8], separation of transaction/consensus blocks [1], hybrid approaches between proof-of-work and proof-of-stake [20], and the use of microblocks [7].

2 Definitions & Model

Notation. Given a sequence Y, we address it using $Y[i]$ to mean the i^{th} element (starting from 0). We use $|Y|$ to denote the length of Y. Negative indices address elements from the end, so $Y[-i]$ is the i^{th} element from the end, and $Y[-1]$ in particular is the last. We use $Y[i{:}j]$ to denote the subarray of Y consisting of the elements indexed from i (inclusive) to j (exclusive). The notation $Y[i{:}]$ means the subarray of Y from i onwards, while $Y[{:}j]$ means the subsequence of Y up to (but not including) j. The notation $\|$ denotes the concatenation of two strings. Given a sequence of strings $(Y_i)_{i\in[n]}$ we denote by $\|_{i\in[n]}Y_i$ the concatenation of all the strings in the sequence, in order. We denote by $\mathsf{Bern}(p)$ the Bernoulli distribution with parameter p, and $\mathsf{Exp}(\lambda)$ the exponential distribution with mean $\frac{1}{\lambda}$. We use $\rightarrow$ to mean implication, and $\Rightarrow$ to mean a logical deduction step in a proof.

The protocols we are interested in are known in the literature as *distributed ledger protocols* (or *logs*, *consensus*, *state machine replication*, or *full atomic com-*

mit channels) and are operated by a set of mutually untrusted parties among which the majority will be assumed to be honest, take as input unordered transactions, and produce as output a safe and live stable *ledger* of transactions (i.e., a sequence of transactions).

Definition 1 (Distributed Ledger Protocol). *A* distributed ledger protocol *is an Interactive Turing Machine (ITM) which exposes the following methods:*

- WRITE(tx)*: Takes user input by accepting some transaction* tx*.*
- READ()*: Produces user output in the form of a* ledger *(a sequence of transactions)*

The distributed ledger protocol is executed by a set of n parties. In a distributed ledger protocol execution, the notation ${}^{P}\boldsymbol{L}_{r}$ denotes the output of READ() invoked on party P at the end of round r. We will call ${}^{P}\boldsymbol{L}_{r}$ simply the *ledger* of P at r, implying that it is the *stable ledger* reported by P. We note that, in our treatment, ledgers are simple sequences of transactions, not blocks.

We denote that ledger ${}^{P_1}\boldsymbol{L}_{r_1}$ is a prefix of ledger ${}^{P_2}\boldsymbol{L}_{r_2}$, using the notation ${}^{P_1}\boldsymbol{L}_{r_1} \preccurlyeq {}^{P_2}\boldsymbol{L}_{r_2}$. When $({}^{P_1}\boldsymbol{L}_{r_1} \preccurlyeq {}^{P_2}\boldsymbol{L}_{r_2}) \vee ({}^{P_2}\boldsymbol{L}_{r_2} \preccurlyeq {}^{P_1}\boldsymbol{L}_{r_1})$ holds, we use the notation ${}^{P_1}\boldsymbol{L}_{r_1} \sim {}^{P_2}\boldsymbol{L}_{r_2}$.

Definition 2 (Safety). *A distributed ledger protocol is* safe *if for any honest parties* P_1, P_2 *and any rounds* r_1, r_2*, it holds that* ${}^{P_1}\boldsymbol{L}_{r_1} \sim {}^{P_2}\boldsymbol{L}_{r_2}$*.*

Definition 3 (Liveness). *A distributed ledger protocol is* live(u) *if for any honest party that attempts to inject a transaction* tx *at round* r*, it holds that* $\mathsf{tx} \in {}^{P}\boldsymbol{L}_{r+u}$ *for all honest parties* P*.*

Definition 4 (Security). *A distributed ledger protocol is* secure *if it is both safe and live*(u)*.*

Blocks & Mining. Similar to Bitcoin, our protocol consists of miners who attempt to find blocks. Each miner locally maintains their *adopted chain* C, which is the best chain so far. A chain is a sequence of blocks beginning with a known genesis block G, considered honest by definition. A block is a triplet of the form $B = (h, x, \mathsf{ctr})$, where $h \in \{0,1\}^{\kappa}$ is a reference to the previous block, $x \in \{0,1\}^{*}$ contains the transactions of the block, and $\mathsf{ctr} \in \mathbb{N}$ is a nonce used to produce proof-of-work. The hash of block B is denoted as $H(B) = H(h \,\|\, x \,\|\, \mathsf{ctr})$. The h is a reference to the previous block hash $H(B')$. We let H be a κ-bit hash function, normalized to the interval $[0, 1)$. [7] Despite the normalization, the hash can be stored in a κ-bit string. When a chain C appears in the execution, we say that block $C[j]$ *extends* block $C[i]$ if $i < j$.

Each honest party attempts to *mine* a block extending the chain they currently have by brute forcing the nonce ctr until $H(B) < T$, where T is the fixed *target*. When this happens, the block is broadcast to all other parties.

[7] The deployed hash function of, e.g., Bitcoin, is SHA-256, which outputs a 256-bit string. The value can be scaled to the interval $[0, 1)$ by dividing by 2^{256}.

Bitcoin Backbone. We analyze the protocol using the model introduced in the Bitcoin Backbone [10] paper. The polynomially bound protocol execution is parametrized by a security parameter $\kappa \in \mathbb{N}$ and orchestrated by an environment $\mathcal{Z}$ which is not unlike Canneti's UC model [3]. The execution commences in discrete rounds $1, 2, \ldots$, and has a total duration of L, polynomial in the security parameter κ. We assume a synchronous communication network with a fixed delay of $\Delta = 1$: If an honest party sends a message to the network at some round r, this message is delivered to all honest parties (including itself) at round $r + 1$. We also assume a static setting, where the protocol is executed by a fixed total number of $n \in \mathbb{N}$ parties, unknown to the honest parties. In the execution, the adversary controls $t < n$ of the parties, and each of the $n - t$ other parties are honest and execute the prescribed Distributed Ledger Protocol. We let the first $1, 2, \ldots, n - t$ parties be the honest parties and the last $n - t + 1, \ldots, n$ parties be the corrupted parties, which may behave arbitrarily. This choice is without loss of generality [10, Proposition 18]. Parties communicate through an unauthenticated network, meaning that the adversary can "spoof" [6] the source address of any message that is delivered. The adversary can also send different messages to different parties in the same round.

Static difficulty. Our analysis is in the *static population* model in which the difficulty and target remain static. In the static model, Bitcoin uses the *longest chain rule* [10], where each block counts for 1 unit. On the contrary, in the real deployment of Bitcoin, the difficulty is dynamically adjusted (the *variable population* model [11]), and the *heaviest chain* is chosen. The scoring in the variable difficulty model makes each block count for $\frac{1}{T} \in (1, \infty)$, where $T \in (0, 1)$ is the nominal target of the block, although T is adjusted from epoch to epoch. In PoEM, we count the *intrinsic work* of each block, which is different from the nominal target T and depends on the value $H(B) < T$, and choose the heaviest chain based on this rule: Each block counts for $-\lg \frac{H(B)}{T}$. In both Bitcoin and PoEM, block validity is the same: A well-formed block is valid if $H(B) < T$. Like Bitcoin, PoEM can also be adapted to work in the variable difficulty setting by adjusting the difficulty depending on the observed block production rate of the system. We perform our analysis in the static population model, and leave the analysis in the variable population model for future work.

We let $H(x)$ be a κ-bit hash function, normalized to the interval $[0, 1)$. One can construct such a normalized hash function by invoking a usual κ-bit hash function (e.g., SHA256) and dividing the output by 2^κ. We model H as a random oracle.

The q-bounded model. Following the tradition of the Bitcoin Backbone [10] paper, during each round, each honest party is allowed to query the random oracle with q different x values. Similarly, the adversary is allowed to query the random oracle with tq different x values.

3 Construction

In the PoEM construction, only the fork choice rule of the original Bitcoin protocol is modified. Honest parties, instead of adopting the longest chain, at the beginning of each round, now adopt the chain with the most intrinsic work.

Algorithm 1 The honest party.

$\mathcal{G}$
$C \leftarrow [\,]$
function CONSTRUCTOR($\mathcal{G}'$)
  $\mathcal{G} \leftarrow \mathcal{G}'$ ▷ Select Genesis Block
  $C \leftarrow [\mathcal{G}]$ ▷ Add Genesis Block to start of chain
  round $\leftarrow 1$
end function
function EXECUTE$_{\text{NET}}(1^\kappa)$
  $\bar{M} \leftarrow$ NET.RECEIVE() ▷ Receive chains from the network
  $C \leftarrow \text{maxvalid}(C \cup \bar{M})$ ▷ Adopt heaviest chain
  $x \leftarrow$ INPUT() ▷ Take all transactions in mempool
  $B \leftarrow \text{PoW}(x, H(C[-1]))$ ▷ Mine a new block
  if $B \neq \perp$ **then** ▷ Successful mining
    NET.BROADCAST($C \,\|\, B$) ▷ Broadcast mined chain
  else
    NET.BROADCAST(C) ▷ Broadcast adopted chain
  end if
  round $\leftarrow$ round+1
end function
function READ
  return $([:-k] \lhd C).x$
end function

Definition 5 (Block Intrinsic Work). *The* intrinsic work *of a block hash* $A \in \{0, \frac{1}{2^\kappa}, \frac{2}{2^\kappa}, \ldots, \frac{2^\kappa-1}{2^\kappa}\}$ *is denoted* $\text{WORK}(A) = \gamma - \lg \frac{A}{T} \in [\gamma, \infty]$*, where* $\gamma \in \mathbb{R}^+$ *is the* bias *parameter of the protocol.*

For genesis, we set $\text{WORK}(G) = 1$ (an arbitrary constant) by convention.

Definition 6 (Chain Intrinsic Work). *The* intrinsic work *of a chain* C *is the sum of the intrinsic work of all blocks in* C*. It is denoted as* $\text{WORK}(C) = \sum_{B \in C} \text{WORK}(H(B))$.

Blockchain notation. For chain C, we write $[\alpha] \lhd C$ to denote the block $C[i]$ of C such that $\text{WORK}(C[:i]) < \alpha \leq \text{WORK}(C[:i+1])$. If $\text{WORK}(C) < \alpha$, then let $[\alpha] \lhd C = \perp$. If α is negative, then $[\alpha] \lhd C$ is defined as the block $C[i]$ of C such that $\text{WORK}(C[i:]) > -\alpha \geq \text{WORK}(C[i+1:])$. We use the slicing notation $[\alpha{:}\beta] \lhd C$ to denote $C[i{:}j]$ where i is the index of $[\alpha] \lhd C$ and j is the index of $[\beta] \lhd C$

in C respectively. The notation $[\alpha:] \lhd C$ means $C[i:]$, and the notation $[:\beta] \lhd C$ means $C[:j]$, where i and j are defined with respect to α and β respectively as above. Any continuous chunk of blocks $C[i:j]$ is called a *subchain* of C. Given a block B, we denote by $B.x$ the sequence of transactions included in B. Given a chain C, we denote by $C.x$ the sequence of transactions in all the blocks of C in order, namely $\|_{B \in C} B.x$.

In Algorithm 1 we show the code of an honest party. First, the party is constructed using the CONSTRUCTOR function (Line 3). In every round, each party is executed by the environment using function EXECUTE (this is an artifact of the lockstep round-based nature of our time model).

Algorithm 2 The Proof-of-Work discovery algorithm

```
1: function PoW_{H,T,q}(x, h)
2:     ctr ←$ {0,1}^κ
3:     for i ← 1 to q do
4:         B ← h ‖ x ‖ ctr
5:         if H(B) < T then
6:             return B
7:         end if
8:         ctr ← ctr + 1
9:     end for
10:    return ⊥
11: end function
```

The honest party begins each round with a certain value stored in his chain C. We say that the honest party *has* chain C at this round. The party calls NET.RECEIVE() to get all the chains from the network (Line 9) and chooses the "best" chain among them. We say that the honest party *adopts* this chain. By ${}^{P}\boldsymbol{C}_{r}$, we denote the chain that was adopted by party P at round r. This comparison for the "best" chain is performed by function MAXVALID in Line 10, and is the single point that we deviate from the original Bitcoin protocol. Next, the honest party attempts to mine a block using the POW function (Line 2), which also remains the same as the original protocol: He repeatedly tries to find a block B that satisfies the POW equation $H(B) < T$, where the target $T \in \{0, \frac{1}{2^\kappa}, \frac{2}{2^\kappa}, \ldots, \frac{2^\kappa - 1}{2^\kappa}\}$ is a small real number in the interval $[0, 1)$. If a block is found, this block is broadcast[8] to the network using function NET.BROADCAST().

We will now analyze the functionality MAXVALID. The method receives as input a set of chains and returns the "best" chain based on a validation and chain adoption rule. The function iterates over all provided chains and first checks their validity in Line 9, using function VALIDATE (Algorithm 4). The

[8]We use the term *broadcast* to mean the unreliable, best-effort anonymous manner of communication between honest parties that guarantees message delivery from one honest party to all other honest parties. This is called *diffuse* in the Backbone series of works.

Algorithm 3 The maxvalid algorithm

```
1: function MAXVALID_G(C̄)
2:     C_max ← [ ]
3:     maxwork ← 0
4:     for C ∈ C̄ do
5:         if ¬VALIDATE_G(C) then
6:             continue
7:         end if
8:         thiswork ← WORK(C)
9:         if thiswork > maxwork then
10:            C_max ← C
11:            maxwork ← thiswork
12:        end if
13:    end for
14:    return C_max
15: end function
```

VALIDATE function remains unchanged compared to Bitcoin. The chains that satisfy the validation rule are compared with one another to find the chain with the most intrinsic work (hereforth "heaviest chain"). Finally, in Line 14, we return the "best" chain C_{max}.

Algorithm 4 The chain validation algorithm remains unchanged. First, in Line 2, we check that the first block of the chain is the genesis block. Then, in Line 8, we check that all blocks satisfy the PoW equation and correctly point to their previous block. State transition validation is excluded for simplicity.

```
1: function VALIDATE_G(C)
2:     if C[0] ≠ G then
3:         return false
4:     end if
5:     ĥ ← H(C[0])
6:     for B ∈ C[1:] do
7:         (h, x, ctr) ← B
8:         if H(B) ≥ T ∨ h ≠ ĥ then
9:             return false                    ▷ Invalid PoW or ancestry
10:        end if
11:        ĥ ← H(B)
12:    end for
13:    return true
14: end function
```

When the time comes to report the stable chain (function READ in Algorithm 1 Line 20), after the function EXECUTE has been called, the honest party removes the unstable part of the chain, namely the last k bits of work from the

chain, and reports the remaining chain as stable. Note that, contrary to Bitcoin, the variable k is measured in *bits of work*, and not in blocks (looking ahead, k will be shown to be polynomial in the security parameter κ, and we will calculate its value in the analysis section).

This concludes the PoEM construction.

4 Experiments

In this section, we report on experimental measurements illustrating the concrete improvements in latency of PoEM as compared to Bitcoin. We implemented a stochastic simulation in 1500 lines of Rust code[9] and used Python's matplotlib to plot the results comparing Bitcoin and PoEM.

We simulate various executions of Bitcoin and PoEM with different parameterizations. In each simulated execution, we fix the block production rate g (honest blocks produced per network delay Δ), the adversarial ratio β (ratio of adversary's mining power divided by the total mining power), and, for PoEM, also the bias parameter γ, and we measure the latency of the system. The latency is defined as the time it takes for a new transaction to become stable in the system (i.e., k-confirmed for a sufficiently large k). The challenge is to measure the latency of the two systems at the same *security level*. As the two systems operate differently, simply taking the same number of blocks (or work) does not constitute a fair comparison.

We measure the time needed from the moment an honest block is mined until the first time any honest party considers the block *stable*, and only for those blocks which eventually do become stable. We use the private mining attack as the adversarial strategy, which was proven [5] to be the best possible attack against Bitcoin in the continuous-time model [14]. In a nutshell, in this strategy, the adversary mines blocks in private on her own chain, whereas the honest parties mine their own blocktree, following the heaviest chain rule (in Bitcoin) or the most intrinsic work rule (in PoEM) respectively. The adversary imposes a network delay of Δ on honest parties. Our simulation begins with all honest parties and the adversary agreeing on a particular genesis block G with no premining having occurred. At that point, the adversary uses the private mining strategy with the goal of conducting a double-spend in a block immediately following G. For a particular execution simulation sample, we determine the *safe* confirmation parameter k that the honest parties *should* have used to avoid this double-spend. Whereas in our experiments, this confirmation parameter is determined retroactively, in the real implementation, the confirmation parameter should be determined prospectively.

We simulate the adversary and the honest parties independently. On the one hand, the adversary mines blocks in a chain without incurring any delay, and therefore without any forks. The adversary's block production rate is $g\frac{\beta}{1-\beta}$. On the other hand, the honest parties incur a network delay of Δ for each mined

[9]The source code is available in the following repository: `https://github.com/commonprefix/poem/tree/main/simulation`

block, and they mine blocks at a rate of g. Because of this delay, the blocks mined by the honest parties form a blocktree.

To save time, instead of wastefully simulating proof of work, we artificially simulate block production using a continuous-time stochastic process. We know that the time between the creation of two honest blocks is exponentially distributed with rate g (because the times of block creation form a Poisson point process [14]). Hence, we take multiple independent samples from $\exp(g)$ to get the interval between each successive honest block production event, thereby giving rise to a Poisson point process of honest block creations (even though these blocks may not necessarily be placed in a chain). We simulate a similar independent Poisson point process for the adversary by sampling from $\exp(g\frac{\beta}{1-\beta})$. These two stochastic processes produce the block creation times of the honest and adversary parties, from which we can determine the blocktree that was constructed taking the delay into account.

In Bitcoin, the work of each block is equal to one, whereas in PoEM, the work of a block is distributed as $\gamma+\exp(\frac{1}{\ln 2})$ (as we show in the Analysis section). Note that these biased exponential samples of *work* are a different, parallel process from the exponential samples of *time intervals* between block creations. Hence, in PoEM's case, we sample from $\gamma+\exp(\frac{1}{\ln 2})$ to get the work of each block in the execution. The creation time of each block and its work are enough to simulate the honest parties' execution and determine the blocktree that was constructed. The honest blocktree is computed as follows: We create a list of *events* pertaining to either the creation of a new honest block, or the arrival of the block to the rest of honest parties Δ later[10]. We sort these events in chronological order and process them one by one. Throughout time, we maintain a single value indicating the cumulative work of the heaviest chain seen so far by all honest parties. For every block, we also keep the cumulative work of the chain ending at that block. At the block creation event, we compute the cumulative work associated with the block as the cumulative work of the heaviest chain seen so far plus the work of the block (1 in Bitcoin, sampled in PoEM). At the block arrival event, we compare the cumulative work of the chain ending at the arrival block with the cumulative work of the heaviest chain seen so far[11]. If it is heavier, we update our best cumulative work value. Every time we update the best cumulative work value, we remember this cumulative work together with the creation timestamp associated with the block that caused the work to increase.

Then, independently we simulate the adversary's execution. Like in the honest execution, we sample from $\exp(g\frac{\beta}{1-\beta})$ to get the interval between the time of successive adversarially produced blocks, where $g\frac{\beta}{1-\beta}$ is the adversary's block production rate. Then, similar to the honest simulation, we set the work of each block to 1 in Bitcoin, and sample from $\gamma+\exp(\frac{1}{\ln 2})$ to get the work of each block in PoEM. Since the adversary has no network delay, all her blocks are chained

[10] Delaying all blocks by the maximum possible delay Δ was proven to be an optimal adversarial strategy [5].

[11] We make the usual [5] simplifying assumption that the number of mining parties $n \to \infty$, meaning that each block is mined by a different party.

in series. Again, we maintain an association of timestamps and cumulative work values for every block.

Having simulated the honest and adversary executions, we can now determine the latency of the system (whether Bitcoin or PoEM). To find this latency, we must find the minimum *safe* confirmation parameter k (where k is denominated in *amount of work* per block, as measured by the respective system). To do this, we determine the *last* point in time when the adversary had a chain with more work than the honest parties, and we record the work k of the honest chain that first surpasses the adversary's chain immediately after. For this execution, a confirmation parameter larger or equal to k would safeguard the protocol from a Common Prefix violation.

To accurately calculate the latency of a given system parameterization (g, β) – and, for PoEM, also γ – we simulate 100,000 such executions and record the minimum safe confirmation parameter k for each execution. We wish to determine the value of k that gives a specific security level – in our case, we want 90% of the executions to be safe from a Common Prefix violation (i.e., 3.32 bits of security[12]). Intuitively, using a common reference security level is what allows us to fairly compare these disparate systems.

In a nutshell, we set k^* to be the minimum confirmation parameter that would safeguard 90% of the executions against a Common Prefix violation. To do this, we collect all the safe k values of the 100,000 Monte Carlo iterations into a list. We take the 20% highest percentile of the list and set k^* to be the mean of these values (i.e., the *expected shortfall* at 20%, noting that the 90% mark falls in the middle of the top 20% percentile).

The latency of the system is calculated as $d = \frac{k^*}{f}$, where f is the average cumulative work increase rate for the honest chains. The latency d is the average time it takes to produce k^* chained honest work in the system (both in Bitcoin and PoEM). To acquire f, we calculate the honest cumulative work increase rate for each execution and take their average across all Monte Carlo iterations. Namely, for each execution, we take the maximum honest cumulative work achieved during the execution's lifetime, and divide it by the time at which it occurred.

We get the optimal system latency for 30 different adversarial ratios β ranging linearly from 0 to 0.4. We explore the latency of the system for 50 different block production rates g increasing geometrically from 0.05 to 85 blocks per network delay. For PoEM, we explore 40 different bias parameters γ starting at 0 and then ranging geometrically from 0.01 to 20. For each tuple (β, g, γ), we run the 100,000 Monte Carlo iterations to obtain the confirmation delay that achieves 90% security (in the expected shortfall sense), and find the minimum delay across all configurations (g, γ) (for PoEM) and g (for Bitcoin) for each β. These optimal latencies are illustrated in Figure 2. For each point in the plot, we ran

[12] While $-\lg_2(10\%) = 3.32$ bits of security are not enough for cryptographic security, we use this as a reference to compare the latency of the two systems. On the contrary, our theoretical analysis shows that PoEM is secure with *overwhelming* probability in the security parameter κ.

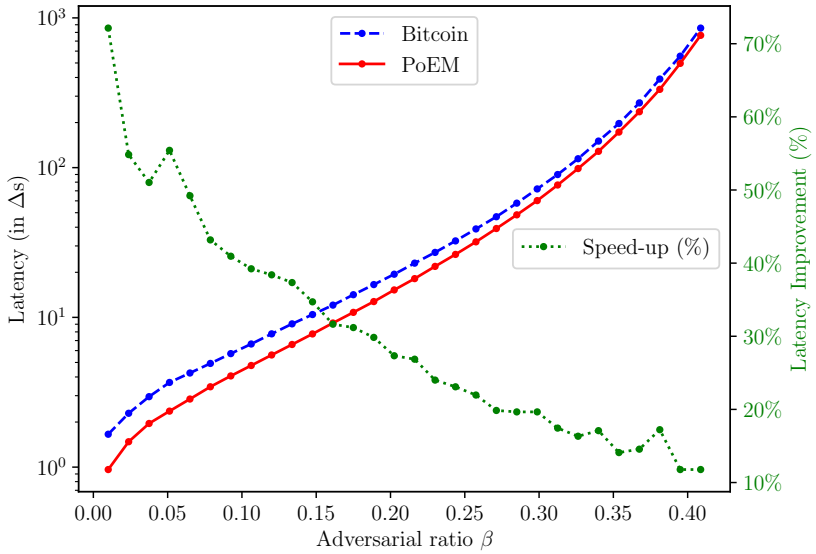


Fig. 2: Bitcoin vs PoEM latency (logarithmic scale) over different adversarial ratios β. The results were obtained by running 200 million simulations per point. For every value of the triplet (β, g, γ), we perform 100,000 Monte Carlo iterations and obtain the latency-optimal parameters (g, γ) for which the latency is illustrated on the vertical axis.

$100{,}000 \times 50 \times 40 = 200{,}000{,}000$ PoEM simulations and $100{,}000 \times 50 = 5{,}000{,}000$ Bitcoin simulations, for a total of 6.15 billion simulation runs across all points in the plot.

We observe that PoEM outperforms Bitcoin in terms of latency across all adversarial ratios β. For small values of β, PoEM has as much as 70% faster confirmation than Bitcoin. The speedup decreases as β increases, but PoEM still outperforms Bitcoin by 11.7% for adversarial ratios around 0.4. For small adversarial presence, the block production rate g of PoEM can be safely increased much more than in Bitcoin. This is illustrated in Figure 3. It is for these larger values of g that PoEM truly shines, utilizing more effectively the work of the forks that appear in the honest blocktree (Figure 1).

Some more detail about how exactly the sampling of adversarial and honest block creation events in the form of the Poisson point process is warranted. We do not re-sample block creation times for each parametrization triplet (β, g, γ) (or (β, g) for Bitcoin respectively). Instead, we sample 100,000 independent Poisson point processes with $\lambda = 1$ for the adversary, and 100,000 additional independent Poisson point processes with $\lambda = 1$ for the honest parties. These are the only timestamp samples we ever take. These Monte Carlo timestamp samples are reused all the same for both Bitcoin and PoEM. As the parameters β and g vary, the existing samples are appropriately scaled (by multiplying their timestamps

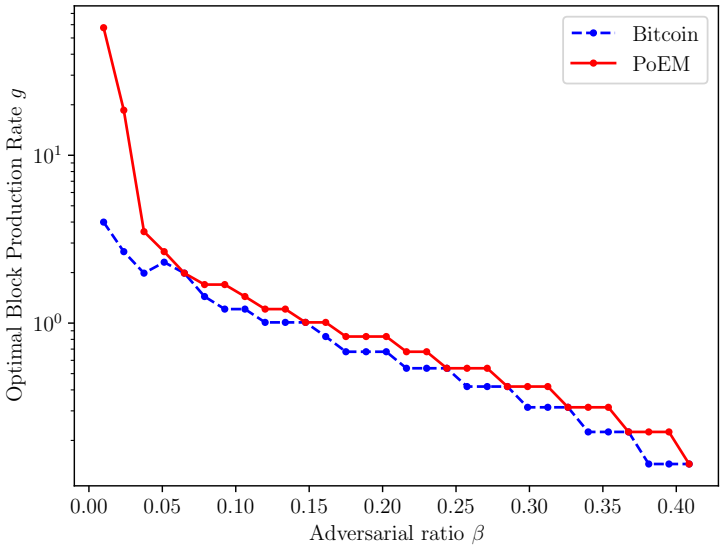


Fig. 3: Bitcoin vs PoEM optimal block production rate g (logarithmic scale) over different adversarial ratios β. The results were obtained by running 100,000 Monte Carlo iterations for each parametrization triplet (β, g, γ), for a total of 200,000,000 simulations per point.

accordingly) to obtain the correct block production rates for the adversary and the honest parties, i.e., the Poisson process of the adversary is adjusted to have a rate of $\lambda = g\frac{\beta}{1-\beta}$ and the Poisson process of the honest parties is adjusted to have a rate of $\lambda = g$. For PoEM in particular, for each block timestamp that appears in our previous samples, we also sample, independently for each block, an additional exponentially distributed (with $\lambda = \frac{1}{\ln 2}$) sample indicating the work of the block. These samples are also re-used across all parameterizations. As the parameter γ varies, the works are adjusted, without resampling, by adding the constant γ to each block's work. Following these time and work adjustments, the honest blocktree is recomputed to account for the delay and different cumulative chain work.

Each execution was concluded when a certain number of (not necessarily chained) blocks had been mined. This number was chosen to be large enough so that k^* was well-defined, namely the executions were chosen to be long enough so that there was always a safe confirmation parameter k. However, as the race between the adversary and the honest parties approaches a tie ($\beta \to 0.5, g \to \infty$), we set an upper bound on the number of blocks to be mined, and if no safe confirmation parameter k is found during the execution lifetime we set k to infinity. In the calculation of the expected shortfall, we discard the executions for which k is infinite.

We note some limitations of our experimental methodology. Firstly, the private attack has been proven optimal for Bitcoin only, and not for PoEM, but in our experiments we have used the same attack for both protocols. Secondly, the optimality of the private attack previously proven [5] is a reduction illustrating that, for a given resilience β, if the private attack is not possible, then no attack is possible. In our experiments, we obtain a safe confirmation parameter k against the private mining attack, but this confirmation parameter has not been shown to be safe against other attacks, even though the private mining attack is optimal when it pertains to resilience. Lastly, there is a discrepancy between the continuous-time model used in our experimental simulations and the discrete-time model used in our theoretical analysis.

Real-world Deployment. In addition to the above simulations, we have implemented and deployed[13] PoEM in a real-world permissionless peer-to-peer setting. The deployment is on a testnet that has been continuously operating for four months. During this period, the network generated 7.5 million blocks and 500 million transactions with participation from 2000 miners from the community, maintaining an average hash rate of 50GH/s using ProgPoW [23]. The miners computed 518 petahashes in 4 months, with the difficulty ($\frac{2^{256}}{T}$) varying between 18 billion and 790 billion.

5 Analysis

To analyze the security of PoEM, we work in the following variant of the Random Oracle model [2], in which the random oracle operates as follows internally.

Definition 7 (Real-Valued Random Oracle). *The* real-valued random oracle H *can be queried with an input value x and returns the value $y[:\kappa]$ as follows. When queried with x for the first time, it samples a real value y uniformly at random from the continuous interval $(0, 1)$, and returns its first κ bits $H(x) = y[:\kappa]$. It then remembers the pair (x, y). We denote by $\overline{\overline{H}}(x)$ this sampled value y. When queried with x for a subsequent time, it returns the first κ bits $H(x) = y[:\kappa]$ of the stored y.*

We note that this definition is equivalent to the standard Random Oracle model, as the real number y is unobservable by any Turing Machine, and the only observable quantity is the κ-bit approximation $y[:\kappa]$. Despite this, in the analysis, we will make use of the real-valued random variable $y = \overline{\overline{H}}(x)$.

Definition 8 (Intrinsic Work). *We define the* intrinsic work *of a real number $A \in [0, 1)$ as $\textsc{work}(A) = \gamma - \lg \frac{A[:\kappa]}{T} = \gamma - \lg \frac{\frac{\lfloor 2^\kappa A \rfloor}{2^\kappa}}{T} \in [\gamma, +\infty]$, where $\gamma \in \mathbb{R}^+$ is the* bias *parameter of the protocol.*

We note that the above definition is a generalization of Definition 5 for $A \in [0, 1)$.

[13] The source code can be found at `https://github.com/dominant-strategies/go-quai/releases/tag/v0.28.2`

Definition 9 (Real Intrinsic Work). *We define the* real intrinsic work *of a real number $A \in [0,1)$ as $\overline{\overline{WORK}}(A) = \gamma - \lg \frac{A}{T} \in [\gamma, +\infty]$, where $\gamma \in \mathbb{R}^+$ is the* bias *parameter of the protocol.*

To simplify the analysis, we will set bias $\gamma = 0$. We observe that, for a block hash $H(x)$, the actual protocol uses $\textsc{work}(H(x))$, which is an approximation of $\overline{\overline{\textsc{work}}}(\overline{\overline{H}}(x))$. Because $T \in \{0, \frac{1}{2^\kappa}, \frac{2}{2^\kappa}, \ldots, \frac{2^\kappa - 1}{2^\kappa}\}$, we have $\textsc{work}(H(B)) \leftrightarrow \overline{\overline{\textsc{work}}}(\overline{\overline{H}}(B))$. We will use the latter value in our analysis. Looking ahead, our goal will be to demonstrate that the small discrepancy between these two values is immaterial to the protocol's output. The cornerstone of this result is stated and proven in the technical *Hash Separation* lemma (Lemma 5) in the appendix. The reason why this real-valued random oracle is useful is that it allows us to borrow tools from analysis to prove statements about the protocol. In particular, the work of a block is distributed as $\exp(\frac{1}{\ln 2})$. The sum of the works of multiple blocks in a chain has a variance that can be bounded using Chernoff bounds that would not be available if we were to use the quantized work.

In a similar vein to the block real intrinsic work, we define the *real* intrinsic work $\overline{\overline{\textsc{work}}}(C)$ of a chain C, which is approximated by its intrinsic work $\textsc{work}(C)$:

Definition 10 (Real Intrinsic Chain Work). *We define the* real intrinsic work *of a chain C as $\overline{\overline{WORK}}(C) = \sum_{B \in C} \overline{\overline{WORK}}(\overline{\overline{H}}(B))$.*

Completely analogously to the chain addressing notation we defined in Section 3, we define the *real* chain addressing notation $[\alpha]\overline{\overline{\lhd}}C$ as follows.
Real blockchain addressing. Let $[\alpha]\overline{\overline{\lhd}}C = C[i]$ where $\overline{\overline{\textsc{work}}}(C[:i]) < \alpha \leq \overline{\overline{\textsc{work}}}(C[:i+1])$. If $\overline{\overline{\textsc{work}}}(C) < \alpha$, then $[\alpha]\overline{\overline{\lhd}}C = \bot$. If $\alpha < 0$, then $[\alpha]\overline{\overline{\lhd}}C = C[i]$ where $\overline{\overline{\textsc{work}}}(C[i:]) < -\alpha \leq \overline{\overline{\textsc{work}}}(C[i+1:])$. Let $[\alpha{:}\beta]\overline{\overline{\lhd}}C = C[i{:}j]$, where i is the index of $[\alpha]\overline{\lhd}C$ and j is the index of $[\beta]\overline{\overline{\lhd}}C$ in C. Let $[\alpha{:}]\overline{\overline{\lhd}}C = C[i{:}]$, and $[{:}\beta]\overline{\overline{\lhd}}C = C[{:}j]$, where i and j are defined with respect to α and β as above.

The following three chain virtues will be used as intermediate stepping stones towards proving the security of the protocol.

Definition 11 (Entropic Growth). *The* Entropic Growth *property of a PoEM execution, parametrized by the growth interval $s \in \mathbb{N}$ and the entropic growth velocity $\tau \in \mathbb{R}^+$, states that for all honest parties P and all rounds $r_1 + s \leq r_2$, the chains C_1, C_2 of P at rounds r_1, r_2 respectively satisfy $\overline{\overline{WORK}}(C_2[|C_1|:]) \geq s\tau$.*

Definition 12 (Existential Entropic Quality). *The* Existential Entropic Quality *property of a PoEM execution, parametrized by the entropic* quality chunk parameter *$\ell \in \mathbb{N}$, states that for all honest parties P and all rounds r, the chain C that P adopts at round r has the property that for every $0 \leq \alpha < \textsc{work}(C) - \ell$, there is at least one honestly generated block in the chain $[\alpha{:}\alpha + \ell]\overline{\overline{\lhd}}C$.*

Definition 13 (Entropic Common Prefix). *The* Entropic Common Prefix *property of a PoEM execution, parametrized by the* common prefix parameter

$k \in \mathbb{N}$ states that for all honest parties P_1, P_2 and all rounds $r_1 \leq r_2$, the chains C_1, C_2 that P_1, P_2 adopt at rounds r_1, r_2 respectively satisfy $[:-k]\overline{\trianglelefteq} C_1 \preccurlyeq C_2$.

The above three properties are proven in Theorems 4, 5, and 6 in the appendix. From these three properties, the safety and liveness of the protocol follow in the next two theorems, which are proven in the appendix.

Theorem 1 (PoEM is Safe). *Typical executions of PoEM are safe.*

Theorem 2 (PoEM is Live). *Typical executions of PoEM are live with parameter* $u = \max(\left\lceil \frac{\ell+2k}{(1-\epsilon)f} \ln 2 \right\rceil, s)$.

The security of the protocol follows from the above two theorems:

Corollary 1 (PoEM is Secure). *PoEM is secure with overwhelming probability.*

6 Conclusion

In this paper, we introduced PoEM, a new fork choice rule, in which each block counts for $\text{WORK}(B) = \gamma - \lg \frac{H(B)}{T}$ instead of the usual $\text{WORK}(B) = \frac{1}{T}$ (Section 3). We illustrated experimentally (Section 4) that PoEM achieves better transaction confirmation latency (roughly 40% improvement). We formally proved the security of PoEM (Corollary 1) in the Bitcoin Backbone model (Section 5). In our proof, we introduced the novel *real-valued random oracle* model (Section 2), which allowed us to use tools from the continuous domain such as the exponential distribution. We showed this model to be closely related to the *discrete random oracle* model (Lemma 5), and believe this new mathematical tooling may be independently useful for the analysis of other protocols. We reported on the production-grade deployment of our protocol, which has an already deployed testnet (Section 4). Our protocol only changes the proof-of-work inequality, and thus is composable with previously proposed improvements for latency and throughput in proof-of-work blockchains. We are hopeful that our modification will be adopted by existing and future proof-of-work protocols in the community.

References

1. V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath. Prism: Deconstructing the Blockchain to Approach Physical Limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, ACM, 1993.

3. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.
4. S. Daveas, K. Karantias, A. Kiayias, and D. Zindros. A Gas-Efficient Superlight Bitcoin Client in Solidity. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 132–144, 2020.
5. A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni. Everything is a race and nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.
6. J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
7. I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59, 2016.
8. M. Fitzi, P. Ga, A. Kiayias, and A. Russell. Parallel Chains: Improving Throughput and Latency of Blockchain Protocols via Parallel Composition. *Cryptology ePrint Archive*, 2018.
9. M. Fitzi, P. Gaži, A. Kiayias, and A. Russell. Ledger combiners for fast settlement. In *Theory of Cryptography Conference*, pages 322–352. Springer, 2020.
10. J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 9057 of *LNCS*, pages 281–310. Springer, Apr 2015.
11. J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In J. Katz and H. Shacham, editors, *Annual International Cryptology Conference*, volume 10401 of *LNCS*, pages 291–323. Springer, Aug 2017.
12. P. Gaži, A. Kiayias, and A. Russell. Tight Consistency Bounds for Bitcoin. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 819–838, 2020.
13. Y. Georghiades, K. Kreder, J. Downing, A. Orwick, and S. Vishwanath. Scalable multi-chain coordination via the hierarchical longest chain rule. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 468–475. IEEE, 2022.
14. D. Guo and L. Ren. Bitcoin's latency–security analysis made simple. *arXiv preprint arXiv:2203.06357*, 2022.
15. K. Karantias, A. Kiayias, and D. Zindros. Compact Storage of Superblocks for NIPoPoW Applications. In P. Pardalos, I. Kotsireas, Y. Guo, and W. Knottenbelt, editors, *Mathematical Research for Blockchain Economy*. Springer International Publishing, 2020.
16. A. Kiayias, N. Lamprou, and A.-P. Stouka. Proofs of proofs of work with sublinear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, Springer, 2016.
17. A. Kiayias, N. Leonardos, and D. Zindros. Mining in Logarithmic Space. CCS '21, New York, NY, USA, 2021. Association for Computing Machinery.
18. A. Kiayias, A. Miller, and D. Zindros. Non-interactive proofs of proof-of-work, 2017.
19. A. Kiayias, A. Polydouri, and D. Zindros. The Velvet Path to Superlight Blockchain Clients. *IACR Cryptology ePrint Archive*, 2020:1122, 2020.

20. E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *25th usenix security symposium (usenix security 16)*, pages 279–296, 2016.
21. K. Kreder and S. Shastry. POEM: Proof of Entropy Minima. *arXiv preprint arXiv:2303.04305*, 2023.
22. A. Miller. The high-value-hash highway. bitcoin forum post, 2012.
23. K.-L. Minehan et al. ProgPoW: A Programmatic Proof-of-Work, 2018. GitHub repository.
24. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
25. R. Pass and E. Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324. ACM, 2017.
26. Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE: Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections. *Cryptology ePrint Archive*, 2016.
27. Y. Sompolinsky, S. Wyborski, and A. Zohar. PHANTOM GHOSTDAG: A Scalable Generalization of Nakamoto Consensus. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 57–70, 2021.
28. Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.
29. Y. Sompolinsky and A. Zohar. PHANTOM: A Scalable BlockDAG protocol. *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.
30. D. Zindros. *Decentralized Blockchain Interoperability*. PhD thesis, University of Athens, Apr 2020.
31. D. Zindros. Soft Power: Upgrading Chain Macroeconomic Policy Through Soft Forks. In *International Conference on Financial Cryptography and Data Security*. Springer, Springer, 2021.
32. D. Zindros. Blockchain Foundations. Unpublished manuscript, 2024.

A Security Analysis with Proofs

We now prove that PoEM is secure. We begin with our central assumption.

Definition 14 (Honest Majority Assumption). *We say that an execution has* honest majority *with* honest advantage parameter $0 < \delta \leq 1$*, if the number t of corrupted parties out of n parties satisfies $t < (1 - \delta)(n - t)$.*

Consider an execution of the PoEM protocol.

We define a random variable $A_{r,i,j}$ as follows. If at round r, the j-th query of (honest or adversarial) party P_i is a valid block B (i.e., $H(B) < T$), then $A_{r,i,j} = \overline{\overline{\text{WORK}}}(\overline{\overline{H}}(B))$. If no valid block is found, $A_{r,i,j} = 0$.

We define $X_r = \max_{j=1}^{q} \max_{i=1}^{n-t} A_{r,i,j}$. If at round r at least one honest party finds a valid block ($X_r > 0$), we say that round r is a *successful round*. We let $f = \Pr[X_r > 0] = 1 - (1 - T)^{q(n-t)} \geq q(n - t)T$. Solving for T we obtain $f = 1 - (1 - T)^{q(n-t)} \Rightarrow 1 - f = (1 - T)^{q(n-t)} \Rightarrow (1 - f)^{\frac{1}{q(n-t)}} = 1 - T \Rightarrow T = 1 - (1 - f)^{\frac{1}{q(n-t)}}$.

In our protocol parametrization, we are free to choose how quickly blocks are produced by honest parties by adjusting the target T parameter, but only some configurations will yield the desired security results. We will set T such that the following condition is satisfied.

Definition 15 (Secure Configuration). *Given an environment which affords $q(n-t)$ queries per round to the honest parties, the secure configuration f of PoEM requires $f = \frac{\delta}{6}$. This is achieved by using the secure target value $T = 1 - (1 - \frac{\delta}{6})^{\frac{1}{q(n-t)}}$.*

We will prove the PoEM protocol is secure if the above configuration is followed.

We let $\overline{X}_r = \sum_{i=1}^{n-t} \sum_{j=1}^{q} A_{r,i,j}$ and

$$\underline{\mathrm{X}}_r = \begin{cases} 0, \text{ if there are no } i,j \text{ with } A_{r,i,j} > 0; \text{ otherwise,} \\ A_{r,i,j}, \text{ where } (i,j) \text{ are the minimum such } (i,j). \end{cases}$$

Observe that $\underline{\mathrm{X}}_r \leq X_r \leq \overline{X}_r$ and $\mathbb{E}[\underline{\mathrm{X}}_r] \leq \mathbb{E}[X_r] \leq \mathbb{E}[\overline{X}_r]$.

We define a random variable Y_r as follows. If at round r exactly one honest party obtains a valid block, then $Y_r = X_r$, and we call r a *convergence opportunity*. Otherwise, $Y_r = 0$.

We define Z_r as the sum of all real intrinsic work generated by all adversarial party queries during round r, namely $Z_r = \sum_{i=n-t+1}^{n} \sum_{j=1}^{q} A_{r,i,j}$.

Given a set of rounds S, we define $X(S) = \sum_{r \in S} X_r$, $\overline{X}(S) = \sum_{r \in S} \overline{X}_r$, $\underline{\mathrm{X}}(S) = \sum_{r \in S} \underline{\mathrm{X}}_r$, $Y(S) = \sum_{r \in S} Y_r$ and $Z(S) = \sum_{r \in S} Z_r$. Observe $\underline{\mathrm{X}}(S) \leq X(S) \leq \overline{X}(S)$.

Lemma 1 (Expectation Bounds). *The following bounds hold.*

1. $\frac{f}{1-f} > Tq(n-t)$
2. $\mathbb{E}[\underline{\mathrm{X}}_r] = \frac{1-(1-T)^{q(n-t)}}{\ln 2} = \frac{f}{\ln 2} > \frac{(1-f)Tq(n-t)}{\ln 2}$
3. $\mathbb{E}[\overline{X}_r] < \frac{f}{1-f}\frac{1}{\ln 2}$
4. $\mathbb{E}[Y_r] > \frac{(1-\frac{\delta}{3})f}{\ln 2}$
5. $\mathbb{E}[Z_r] = \frac{tqT}{\ln 2} < \frac{t}{n-t} \cdot \frac{f}{1-f} \cdot \frac{1}{\ln 2} < \left(1 + \frac{\delta}{2}\right) \frac{t}{n-t} \cdot \frac{f}{\ln 2}$
6. $\mathbb{E}[Z_r] < \mathbb{E}[X_r]$

Proof. Observe that $A_{r,i,j}$ can be expressed in the form $A_{r,i,j} = C_{r,i,j} W_{r,i,j}$, with independent boolean random variable $C_{r,i,j} \sim \mathsf{Bern}(T)$ indicating whether the query was successful and real random variable $W_{r,i,j} \sim \mathsf{Exp}(\ln 2)$ measuring the real work of the block found. Concerning expectations, $\mathbb{E}[W_{r,i,j}] = \frac{1}{\ln 2}$, and, furthermore, $\mathbb{E}[A_{r,i,j}] = \mathbb{E}[A_{r,i,j} | C_{r,i,j} = 0] \Pr[C_{r,i,j} = 0] + \mathbb{E}[A_{r,i,j} | C_{r,i,j} = 1] \Pr[C_{r,i,j} = 1] = \mathbb{E}[A_{r,i,j} | C_{r,i,j} = 1] \Pr[C_{r,i,j} = 1] = \frac{T}{\ln 2}$. The following bounds are similar to [10].

Bounds for f. We note that $\frac{f}{1-f} = \frac{1-(1-T)^{q(n-t)}}{(1-T)^{q(n-t)}} = (1-T)^{-q(n-t)} - 1 > (1+T)^{q(n-t)} - 1 > Tq(n-t)$. Here, the penultimate inequality stems from

$(1-T)^{-q(n-t)} > (1+T)^{q(n-t)} \Leftarrow (1-T)^{-1} > 1+T \Leftarrow 1-T^2 < 1 \Leftarrow T > 0$. The last inequality stems from Bernoulli's inequality, namely $(1+x)^r \geq 1+rx$ for integer $r \geq 1$ and real $x \geq -1$.

Bounds for $\mathbb{E}[X]$. The expectation $\mathbb{E}[\underline{X}_r] = \frac{1-(1-T)^{q(n-t)}}{\ln 2}$ follows from fact that $\underline{X}_r \sim \mathsf{Bern}(1-(1-T)^{q(n-t)})\mathsf{Exp}(\ln 2)$. The bound on $\mathbb{E}[\underline{X}_r]$ follows from the previous bound on f. The expectation $\mathbb{E}[\overline{X}_r] = \frac{Tq(n-t)}{\ln 2} < \frac{f}{1-f}\frac{1}{\ln 2}$ follows from the fact that $\overline{X}_r$ is the sum of $q(n-t)$ independent random variables distributed as $\mathsf{Bern}(T)\mathsf{Exp}(\ln 2)$ and the above bounds on f.

Bounds for $\mathbb{E}[Y]$. The probability of a convergence opportunity is $(n-t)(1-(1-T)^q)(1-T)^{q(n-t-1)} \geq Tq(n-t)(1-T)^{q(n-t)-1} > Tq(n-t)(1-(q(n-t)-1)T) > Tq(n-t)(1-Tq(n-t))$. The first expression is the binomial probability that exactly one, among $n-t$, honest party is successful; the second is the binomial probability that exactly one, among $q(n-t)$, honest query is successful, which implies that exactly one honest party was successful. The penultimate inequality is by Bernoulli's inequality, namely $(1+x)^r \geq 1+rx$ for integer $r \geq 1$ and real $x \geq -1$.

We have $Y_r \sim \mathsf{Bern}((n-t)(1-(1-T)^q)(1-T)^{q(n-t-1)})\mathsf{Exp}(\ln 2)$, therefore, $\mathbb{E}[Y_r] > \frac{Tq(n-t)(1-Tq(n-t))}{\ln 2} \geq \frac{f(1-f)}{\ln 2} > \frac{(1-\frac{\delta}{3})f}{\ln 2}$. For the inequality concerning $\mathbb{E}[Y_r]$, the derivation is analogous to [10].

Bounds for $\mathbb{E}[Z]$. The expectation $\mathbb{E}[Z_r] = \frac{tqT}{\ln 2}$ follows from the fact that Z_r is distributed as a sum of tq independent samples distributed as $\mathsf{Bern}(T)\mathsf{Exp}(\ln 2)$. For the bound, we have $\mathbb{E}[Z_r] < \frac{t}{n-t}\frac{f}{1-f}\frac{1}{\ln 2} < \left(1+\frac{\delta}{2}\right)\frac{t}{n-t}\cdot\frac{f}{\ln 2}$ using an analysis completely analogous to the one in [10].

For $\mathbb{E}[Z_r] < \mathbb{E}[X_r]$, we have $\mathbb{E}[Z_r] < \mathbb{E}[X_r] \Leftarrow \mathbb{E}[Z_r] < \mathbb{E}[\underline{X}_r] \Leftarrow \frac{tqT}{\ln 2} < \frac{(1-f)Tq(n-t)}{\ln 2} \Leftarrow \frac{t}{n-t} < 1-f \Leftarrow 1-\delta < 1-f \Leftarrow f < \delta$, which follows from the secure configuration.

Definition 16 (Causality). *An execution is* causal *if no block (directly or indirectly) extends one which is computed at a later or the same random oracle query.*

Definition 17 (Hash Separation). *An execution has Hash Separation if for all two (adversarial or honest) chains C_1, C_2 appearing in the execution, if $\overline{\overline{WORK}}(C_1) < \overline{\overline{WORK}}(C_2)$, then $WORK(C_1) < WORK(C_2)$.*

Definition 18 (PoEM Typical Execution). *An execution of PoEM is (ϵ, λ)-*typical *(or just typical), for $\epsilon \in (0,1)$ and integer $\lambda > 4$, if for any set S of at least λ consecutive rounds, the following hold.*

- $$(1-\epsilon)\,\mathbb{E}[\underline{X}(S)] < X(S) < (1+\epsilon)\,\mathbb{E}[\overline{X}(S)] \quad (1)$$
- $$(1-\epsilon)\,\mathbb{E}[Y(S)] < Y(S) \quad (2)$$
- $$Z(S) < (1+\epsilon)\,\mathbb{E}[Z(S)] \quad (3)$$
- *It is causal.*
- *It has hash separation.*

In our analysis, we will let $\epsilon = \frac{\delta}{6}$. If the desired maximum probability of failure is made concrete, this ϵ, together with the concrete probabilities later calculated in Theorem 3, will determine the concrete value of λ, from which the rest of the concrete protocol parameters follow. In particular, the value k for the ledger stabilization rule is determined by λ and f, and λ can be calculated from ϵ, whereas both f and ϵ can be determined from the desired acceptable honest advantage δ.

We will now prove that typical executions occur with overwhelming probability. Towards this purpose, we will need a couple of auxiliary lemmas.

In the following arguments, we connect the *real valued* random oracle, evaluated using the $\overline{\overline{\text{WORK}}}(\overline{\overline{H}}(B)) \in \mathbb{R}^+$ function (an ideal quantity unobservable by any Turing Machine, as it cannot process real-valued inputs), and its κ-bit *discrete approximation* $\text{WORK}(H(B))$ (observable by a Turing Machine). We show the difference between these two quantities is immaterial for polynomially bound computations, namely they notably diverge only with negligible probability. This connection between real-valued and discrete-valued random variables will allow us to conduct our analysis using *continuous* random variables and, in particular, random variables distributed according to the exponential distribution. These distributions lend themselves to easier tools than conducting a cumbersome analysis in the discrete domain; for instance, the sum of i.i.d. exponentially distributed variables is the gamma distribution. The following lemmas that translate between the continuous and discrete worlds will allow us to later utilize our continuous results in the discrete realization of the protocol.

First, we make a few observations about the relationship between the real and the discrete work of blocks and chains. Observe that for hash input A, we have $H(A) \leq \overline{\overline{H}}(A)$ and for block B we have $\text{WORK}(H(B)) \geq \overline{\overline{\text{WORK}}}(\overline{\overline{H}}(B))$, and for blocks B_1, B_2 we have $\overline{\overline{\text{WORK}}}(\overline{\overline{H}}(B_1)) \geq \overline{\overline{\text{WORK}}}(\overline{\overline{H}}(B_2)) \rightarrow \text{WORK}(H(B_1)) \geq \text{WORK}(H(B_2))$.

Furthermore, discretizing real works preserves the order of blocks in the following fashion.

Lemma 2. *For all $A, B \in (0,1)$, it holds that $\overline{\overline{\text{WORK}}}(A) \geq \overline{\overline{\text{WORK}}}(B) \rightarrow \text{WORK}(A) \geq \text{WORK}(B)$.*

Proof.

$$\overline{\overline{\text{WORK}}}(A) \geq \overline{\overline{\text{WORK}}}(B) \Rightarrow -\lg A \geq -\lg B \Rightarrow \lg A \leq \lg B$$
$$\Rightarrow A \leq B \Rightarrow 2^\kappa A \leq 2^\kappa B \Rightarrow \lfloor 2^\kappa A \rfloor \leq \lfloor 2^\kappa B \rfloor \Rightarrow \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \leq \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa}$$
$$\Rightarrow \lg \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \leq \lg \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa} \Rightarrow -\lg \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \geq -\lg \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa}$$
$$\Rightarrow \text{WORK}(A) \geq \text{WORK}(B)$$

□

Showing that the order of *chains* is preserved under discretization is a bit more involved, and we will work towards it next. Towards this, we observe that the discrete work of a block is close to its real work.

Lemma 3 (Block Work Approximation). *In a PoEM execution, consider the event CLOSE that all blocks B have $\textsc{work}(B) - \overline{\overline{\textsc{work}}}(B) < 2^{-\kappa/2}$. The probability $\Pr[CLOSE]$ is overwhelming in κ.*

Proof. Consider the event E in which for all blocks B it holds that $\overline{\overline{H}}(B) > 2^{-(\kappa/2-2)}$. Let us calculate the probability of $\neg E$. For $\neg E$ to happen, at least one block must have $\overline{\overline{H}}(B) \leq 2^{-(\kappa/2-2)}$. For any block B, it holds that $\Pr[\overline{\overline{H}}(B) \leq 2^{-(\kappa/2-2)}] = 2^{-(\kappa/2-2)}$ (from the uniform distribution of $\overline{\overline{H}}(B)$ in the interval $(0,1)$ due to it being a real-valued random oracle). Since there are at most nqL blocks in the execution, by applying a union bound, we have $\Pr[\neg E] = \Pr[\exists B : \overline{\overline{H}}(B) \leq 2^{-(\kappa/2-2)}] \leq \sum_B \Pr[\overline{\overline{H}}(B) \leq 2^{-(\kappa/2-2)}] \leq nqL2^{-(\kappa/2-2)}$, which is negligible in κ, so E happens with overwhelming probability.

Consider a block B of the execution, conditioned on the event E. Then

$$
\begin{aligned}
&\textsc{work}(B) - \overline{\overline{\textsc{work}}}(B) = -\lg H(B) - (-\lg \overline{\overline{H}}(B)) \\
&\overset{(i)}{<} \lg \overline{\overline{H}}(B) - \lg\left(\overline{\overline{H}}(B) - 2^{-\kappa}\right) \overset{(ii)}{<} \lg 2^{-(\kappa/2-2)} - \lg\left(2^{-(\kappa/2-2)} - 2^{-\kappa}\right) \\
&= -\left(\frac{\kappa}{2} - 2\right) - \lg(2^{-(\kappa/2-2)}(1 - 2^{-(\kappa/2+2)})) \\
&= -\left(\frac{\kappa}{2} - 2\right) + \left(\frac{\kappa}{2} - 2\right) - \lg\left(1 - 2^{-(\kappa/2+2)}\right) = -\lg\left(1 - 2^{-(\kappa/2+2)}\right) \\
&= -\ln\left(1 - 2^{-(\kappa/2+2)}\right) \cdot \frac{1}{\ln 2} \overset{(iii)}{\leq} -\frac{-2^{-(\kappa/2+2)}}{1 - 2^{-(\kappa/2+2)}} \cdot \frac{1}{\ln 2} \\
&\leq \frac{2^{-(\kappa/2+2)}}{1 - \frac{1}{2}} \cdot \frac{1}{\ln 2} = 2^{-(\kappa/2+1)} \cdot \frac{1}{\ln 2} < 2^{-\kappa/2}\,.
\end{aligned}
$$

Inequality (i) stems from the fact that $\overline{\overline{H}}(B) - 2^{-\kappa} < H(B) \leq \overline{\overline{H}}(B)$ and the monotonicity of lg. Inequality (ii) stems from the fact that the function $\lg x - \lg(x - 2^{-\kappa}) = \lg \frac{x}{x-2^{-\kappa}}$ is decreasing for $x > 2^{-\kappa}$, remembering our conditioning on $\overline{\overline{H}}(B) > 2^{-(\kappa/2-2)} > 2^{-\kappa}$. Inequality (iii) stems from the standard logarithm inequality $\ln(1+x) \geq \frac{x}{1+x}$ for all $x > -1$. □

The discrete work of a chain is also close to the real work of a chain.

Corollary 2 (Chain Work Approximation). *In a PoEM execution, the probability that all subchains C^* have $\textsc{work}(C^*) - \overline{\overline{\textsc{work}}}(C^*) < Lqn2^{-\kappa/2}$ is overwhelming in κ.*

Proof. Conditioned on the overwhelming event of Lemma 3, for all subchains C^* it holds that

$$
\begin{aligned}
\textsc{work}(C^*) - \overline{\overline{\textsc{work}}}(C^*) &= \sum_{B \in C^*} \textsc{work}(B) - \overline{\overline{\textsc{work}}}(B) \\
< \sum_{B \in C^*} 2^{-\kappa/2} &= |C*|2^{-\kappa/2} \leq Lqn2^{-\kappa/2}\,.
\end{aligned}
$$

□

We are now ready to prove a technical lemma which shows that works of chains do not fall dangerously close to each other.

Lemma 4. *Consider a PoEM execution $\mathcal{E}$ with n parties, q queries per round per party, and total lifetime L. Consider the j-th random oracle query in this execution. If the query is successful, let B indicate its produced block, let $\overline{\overline{w}} = \overline{\overline{WORK}}(B), w = \textsc{work}(B)$, and let C be the chain it extends, let $\overline{\overline{w}}_1 = \overline{\overline{WORK}}(C), w_1 = \textsc{work}(C)$, and $\overline{\overline{w}}'_1 = \overline{\overline{WORK}}(CB), w'_1 = \textsc{work}(CB)$. Consider any other chain C_i that appears in the execution, and let $\overline{\overline{w}}_2 = \overline{\overline{WORK}}(C_i), w_2 = \textsc{work}(C_i)$. Let $BADRANGE_{j,i}$ denote the event that both $\overline{\overline{w}}_1 < \overline{\overline{w}}_2$, and, furthermore, either $\overline{\overline{w}}_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq \overline{\overline{w}}_1 + \overline{\overline{w}} < \overline{\overline{w}}_2$ or $\overline{\overline{w}}_2 < \overline{\overline{w}}_1 + \overline{\overline{w}} \leq \overline{\overline{w}}_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}$. Let $BADRANGE$ denote the event that there exists a random oracle query j and a chain C_i in the execution such that $BADRANGE_{j,i}$. The probability $\Pr[BADRANGE]$ is negligible in κ.*

Proof. If the j-th query does take place, its $\overline{\overline{w}}$ is distributed as $\mathsf{Exp}(\ln 2)$, so for every other chain C_i in the execution for which $\overline{\overline{w}}_1 < \overline{\overline{w}}_2$ we have

$$\begin{aligned}
\Pr[\overline{\overline{w}}_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq \overline{\overline{w}}_1 + \overline{\overline{w}} < \overline{\overline{w}}_2 | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] = \\
\Pr[\overline{\overline{w}}_2 - \overline{\overline{w}}_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq \overline{\overline{w}} < \overline{\overline{w}}_2 - \overline{\overline{w}}_1 | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] = \\
(1 - 2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)}) - (1 - 2^{-\left(\overline{\overline{w}}_2 - \overline{\overline{w}}_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}\right)}) = \\
2^{-\left(\overline{\overline{w}}_2 - \overline{\overline{w}}_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}\right)} - 2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)} = \\
2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)}(2^{\frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}} - 1) \leq 2^{\frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}} - 1 < \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\,.
\end{aligned}$$

The second relation is from the cumulative distribution function of the exponential distribution; the fifth relation is from the conditioning on $\overline{\overline{w}}_1 < \overline{\overline{w}}_2$, and the last relation is from Lemma 12, noting that $0 < \frac{nqL+1}{2^{\kappa/2}} < 1$.

Similarly, for the other direction,

$$\begin{aligned}
\Pr[\overline{\overline{w}}_2 < \overline{\overline{w}}_1 + \overline{\overline{w}} \leq \overline{\overline{w}}_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] = \\
\Pr[\overline{\overline{w}}_2 - \overline{\overline{w}}_1 < \overline{\overline{w}} \leq \overline{\overline{w}}_2 - \overline{\overline{w}}_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] = \\
(1 - 2^{-\left(\overline{\overline{w}}_2 - \overline{\overline{w}}_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\right)}) - (1 - 2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)}) = \\
2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)} - 2^{-\left(\overline{\overline{w}}_2 - \overline{\overline{w}}_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\right)} = \\
2^{-(\overline{\overline{w}}_2 - \overline{\overline{w}}_1)}(1 - 2^{-\frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}}) \leq 1 - 2^{-\frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}} < \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\,.
\end{aligned}$$

Consequently,

$$\begin{aligned}
&\Pr[\text{BADRANGE}_{j,i}] = \Pr[\text{BADRANGE}_{j,i}|\overline{\overline{w}}_1 < \overline{\overline{w}}_2]\Pr[\overline{\overline{w}}_1 < \overline{\overline{w}}_2] \\
\leq &\Pr[\text{BADRANGE}_{j,i}|\overline{\overline{w}}_1 < \overline{\overline{w}}_2] \\
= &\Pr[\overline{\overline{w}}_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq \overline{\overline{w}}_1 + \overline{\overline{w}} < \overline{\overline{w}}_2 | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] + \\
&\Pr[\overline{\overline{w}}_2 < \overline{\overline{w}}_1 + \overline{\overline{w}} \leq \overline{\overline{w}}_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | \overline{\overline{w}}_1 < \overline{\overline{w}}_2] \\
= &2\frac{nqL+1}{2^{\kappa/2}}
\end{aligned}$$

Applying a union bound over all the queries j and chains i of the execution, we obtain $\Pr[\text{BADRANGE}] \leq 2(nqL)^2 \cdot \frac{nqL+1}{2^{\kappa/2}}$, which is negligible in κ. □

Lemma 5 (Hash Separation). *A causal execution of PoEM has Hash Separation except with negligible probability in κ.*

Proof. Consider a causal execution of PoEM for which the event CLOSE of Lemma 3 and the event ¬BADRANGE of Lemma 4 both hold. Observe that the statement of Corollary 2 holds in this conditioning. From the two lemmas we know Pr[CLOSE] and Pr[¬BADRANGE] are both overwhelming, therefore Pr[CLOSE ∧ ¬BADRANGE] is overwhelming. Conditioned on this event, we will show that the desired statement holds with probability 1.

Let HS be the event that Hash Separation holds. Let HS_j denote the predicate that HS holds for all chains appearing before, or at, the j-th random oracle query, with $j = 0$ indicating the beginning of the execution. We will use induction on j to show that for all $0 \leq j \leq Lnq$, HS_j holds. We know that HS_0 always holds by definition.

Now, consider the j-th random oracle query and suppose HS_{j-1} holds. If the query was unsuccessful, then HS_j holds, and we are done. Otherwise, let C_1 be the chain that the j-th random oracle query extends, let B be the block mined on it, let $C_1' = C_1 B$, and let $\overline{\overline{w}} = \overline{\overline{\text{WORK}}}(B), \overline{\overline{w}}_1 = \overline{\overline{\text{WORK}}}(C_1), \overline{\overline{w}}_1' = \overline{\overline{\text{WORK}}}(C_1')$ and w, w_1, w_1' be the respective discrete works. Consider any other chain C_2 with work $\overline{\overline{w}}_2 = \overline{\overline{\text{WORK}}}(C_2)$ and discrete work w_2 that has already appeared in the execution, and consider the undesirable event FLIP_{C_1,C_2} that $\overline{\overline{w}}_1' < \overline{\overline{w}}_2 \wedge w_1' \geq w_2$ or $\overline{\overline{w}}_1' > \overline{\overline{w}}_2 \wedge w_1' \leq w_2$. If $\overline{\overline{w}}_1 \geq \overline{\overline{w}}_2$, then, because $\overline{\overline{w}} > 0$, therefore $\overline{\overline{w}}_1 + \overline{\overline{w}} > \overline{\overline{w}}_2$ and hence $\overline{\overline{w}}_1' > \overline{\overline{w}}_2$. Additionally, by HS_{j-1} we have $w_1 > w_2$, therefore $w_1 + w > w_2$, and $w_1' > w_2$. From this, it follows that $\overline{\overline{w}}_1 \geq \overline{\overline{w}}_2$ yields $\neg\text{FLIP}_{C_1,C_2}$. Thus, it suffices to only consider the situation where $\overline{\overline{w}}_1 < \overline{\overline{w}}_2$.

Case 1: $\overline{\overline{w}}_1 + \overline{\overline{w}} < \overline{\overline{w}}_2$. From the conditioning on ¬BADRANGE, we have $\overline{\overline{w}}_1 + \overline{\overline{w}} < \overline{\overline{w}}_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}$, therefore $w_1 - \frac{nqL}{2^{\kappa/2}} + \overline{\overline{w}} < \overline{\overline{w}}_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \Rightarrow w_1 + \overline{\overline{w}} < \overline{\overline{w}}_2 - \frac{1}{2^{\kappa/2}} \Rightarrow w_1 + w - \frac{1}{2^{\kappa/2}} < \overline{\overline{w}}_2 - \frac{1}{2^{\kappa/2}} \Rightarrow w_1 + w < \overline{\overline{w}}_2 \Rightarrow w_1' < \overline{\overline{w}}_2 \leq w_2$. The first inequality is obtained from the conditioning on CLOSE, noting that $w_1 - \frac{nqL}{2^{\kappa/2}} < \overline{\overline{w}}_1$ follows from $w_1 - \overline{\overline{w}}_1 < Lqn2^{-\kappa/2}$ (Corollary 2). The third inequality is also

obtained from the conditioning on CLOSE, noting that $w - \frac{1}{2^{\kappa/2}} < \overline{\overline{w}}$ follows from $w - \overline{\overline{w}} < 2^{-\kappa/2}$ (Lemma 3). It follows that $\neg\text{FLIP}_{C_1,C_2}$.

Case 2: $\overline{\overline{w}}_1 + \overline{\overline{w}} > \overline{\overline{w}}_2$. From the conditioning on $\neg$BADRANGE, we have $\overline{\overline{w}}_1 + \overline{\overline{w}} > \overline{\overline{w}}_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} > \overline{\overline{w}}_2 + \frac{nqL}{2^{\kappa/2}}$, therefore $\overline{\overline{w}}_1 + \overline{\overline{w}} > w_2 - \frac{nqL}{2^{\kappa/2}} + \frac{nqL}{2^{\kappa/2}} \Rightarrow \overline{\overline{w}}_1 + \overline{\overline{w}} > w_2 \Rightarrow w_1 + w > w_2 \Rightarrow w'_1 > w_2$. Again, it follows that $\neg\text{FLIP}_{C_1,C_2}$.

From this and HS_{j-1} it follows that HS_j holds. Therefore, by induction, HS_{Lnq} holds, and hence HS holds. Since our conditioning was on an overwhelming event, the lemma follows. □

Corollary 3 (Approximate Fork Choice). *In Hash Separated executions of PoEM, for any two chains C_1, C_2 it holds that $\textsc{work}(C_1) < \textsc{work}(C_2) \rightarrow \overline{\overline{\textsc{work}}}(C_1) < \overline{\overline{\textsc{work}}}(C_2)$.*

Proof. Suppose towards a contradiction $\textsc{work}(C_1) < \textsc{work}(C_2)$, but $\overline{\overline{\textsc{work}}}(C_1) \geq \overline{\overline{\textsc{work}}}(C_2)$. From Hash Separation, it follows that $\textsc{work}(C_1) \geq \textsc{work}(C_2)$, which is a contradiction. □

Theorem 3 (Typicality). *An execution of duration L of PoEM is (ϵ, λ)-typical with probability $1 - e^{-\Omega(\lambda - \log L)} - e^{-\Omega(\kappa - \log L)}$, namely, overwhelming in λ and κ.*

Proof. For each S with $|S| = \lambda$,

$$\begin{aligned}
\Pr[X(S) < (1-\epsilon)\,\mathbb{E}[\underline{X}(S)]] \leq \\
\Pr[\underline{X}(S) < (1-\epsilon)\,\mathbb{E}[\underline{X}(S)]] \leq e^{-\Omega(\lambda)}\,. \\
\Pr[X(S) > (1+\epsilon)\,\mathbb{E}[\overline{X}(S)]] \leq \\
\Pr[\overline{X}(S) > (1+\epsilon)\,\mathbb{E}[\overline{X}(S)]] \leq e^{-\Omega(\lambda)}\,. \\
\Pr[Y(S) < (1-\epsilon)\,\mathbb{E}[Y(S)]] \leq e^{-\Omega(\lambda)}\,. \\
\Pr[Z(S) > (1+\epsilon)\,\mathbb{E}[Z(S)]] \leq e^{-\Omega(\lambda)}\,.
\end{aligned}$$

The $e^{-\Omega(\lambda)}$ bounds are obtained by applying Lemma 13 to each of the random variables $\underline{X}(S), \overline{X}(S), Y(S)$ and $Z(S)$, each of which is the sum of $\Theta(\lambda)$ i.i.d. random variables distributed according to $\mathsf{Bern}(p) \times \mathsf{Exp}(\ln 2)$ for some respective $p \in (0,1)$. Applying a union bound for all S (of which there are $L - \lambda + 1$), we obtain that typicality Eq. 1, Eq. 2 and Eq. 3 hold with probability $1 - e^{-\Omega(\lambda) + \ln L}$. If typicality bounds hold for all S with $|S| = \lambda$, then they hold for all S with $|S| \geq \lambda$.

The probability bound for causality follows from the stochastic nature of the Random Oracle and is proven in [10].

Lastly, Hash Separation follows from Lemma 5. □

Definition 19 (Block Work Interval). *A block B of chain C has* work interval $I(B) = \{\xi \geq 0 : [\xi]\overline{\overline{\lhd}}C = B\}$.

Lemma 6 (Entropic Pairing Lemma). *Consider a typical execution of PoEM. Suppose a block B of a chain C with work interval $I(B)$ was computed by an honest party in a convergence opportunity. For every $\xi \in I(B)$ and every chain C' of the execution, block $B' = [\xi] \lhd C'$ is either B or adversarial, as long as $B' \neq \perp$.*

Proof. Consider an execution as in the statement and suppose, towards a contradiction, that block B' is not B and is honestly computed. Since B was computed in a convergence opportunity, B and B' cannot have been computed in the same round. Let r be the earliest round on which B or B' was computed, and C be the chain whose tip this block is. Since it was computed by an honest party, at round $r+1$, every other honest party receives a chain with real work greater or equal to ξ.

Claim: Every block computed after round r will be extending a chain with real work at least ξ. To see this, consider a chain C^* that an honest party is extending after r. Since the party has adopted C^*, by the heaviest chain rule, $\textsc{work}(C^*) \geq \textsc{work}(C)$. By Hash Separation, $\overline{\overline{\textsc{work}}}(C^*) \geq \overline{\overline{\textsc{work}}}(C) \geq \xi$.

If B is computed after round r, it holds that $\xi \notin I$ (noting that $\overline{\overline{\textsc{work}}}(B) > 0$). If B' is computed after round r, it holds that $B' \neq [\xi] \lhd C'$. Both lead to a contradiction. □

Lemma 7 (Entropic Chain Growth Lemma). *Suppose that at round r_1 an honest party P_1 has a chain which has real work $\overline{\overline{w}}$. Then, at round $r_2 \geq r_1$, every honest party P_2 adopts a chain which has real work at least $\overline{\overline{w}} + \sum_{r=r_1}^{r_2-1} X_r$.*

Proof. By induction on r_2. For the inductive base ($r_2 = r_1$), observe that if at round r_1 party P_1 has a chain C of work $\overline{\overline{w}}$, then P_1 broadcasted C at the end or round $r_1 - 1$. Party P_2 receives C at round r_1. Consider the chain C_2 that P_2 adopts at r_1. Due to the heaviest chain rule, $\textsc{work}(C_2) \geq \textsc{work}(C)$, therefore, by Hash Separation, $\overline{\overline{\textsc{work}}}(C_2) \geq \overline{\overline{w}}$, and the statement follows.

For the inductive step, note that by the inductive hypothesis, every honest party has adopted a chain of real work at least $\overline{\overline{w}}' = \overline{\overline{w}} + \sum_{r=r_1}^{r_2-2} X_r$ at round $r_2 - 1$. When $X_{r_2-1} = 0$ the statement follows directly, so assume $X_{r_2-1} > 0$. Observe that an honest party P_3 successfully queried the random oracle at round $r_2 - 1$ and obtained a chain C_3 of real work at least $\overline{\overline{w}}' + X_{r_2-1}$ and broadcasted it to the network. At round r_2, party P_2 receives C_3 and adopts a chain C_2, with $\textsc{work}(C_2) \geq \textsc{work}(C_3)$ due to the heaviest chain rule. By Hash Separation, $\overline{\overline{\textsc{work}}}(C_2) \geq \overline{\overline{\textsc{work}}}(C_3) \geq \overline{\overline{w}}' + X_{r_2-1} = \overline{\overline{w}} + \sum_{r=r_1}^{r_2-1} X_r$. □

Lemma 8 (Typical Bounds). *In typical PoEM executions, for any set S of at least λ consecutive rounds, it holds that:*

1. $Z(S) < \frac{t}{n-t} \cdot \frac{f}{1-f} \cdot \frac{|S|}{\ln 2} + \epsilon f \frac{|S|}{\ln 2} \leq (1 - \frac{2\delta}{3}) f \frac{|S|}{\ln 2}$.
2. $Z(S) < \left(1 + \frac{\delta}{2}\right) \frac{t}{n-t} X(S) + \frac{\epsilon f |S|}{\ln 2}$.
3. $Z(S) < Y(S)$.

Proof. **Proposition 1.**

$$\begin{aligned} Z(S) &< (1+\epsilon)\,\mathbb{E}[Z(S)] = (1+\epsilon)\,\mathbb{E}[Z_r]|S| \\ &= \mathbb{E}[Z_r]|S| + \epsilon\,\mathbb{E}[Z_r]|S| \\ &< \frac{t}{n-t}\cdot\frac{f}{1-f}\cdot\frac{|S|}{\ln 2} + \epsilon\frac{t}{n-t}\cdot\frac{f}{1-f}\cdot\frac{|S|}{\ln 2} \\ &< \frac{t}{n-t}\cdot\frac{f}{1-f}\cdot\frac{|S|}{\ln 2} + \epsilon f\frac{|S|}{\ln 2} \\ &= \left(\frac{t}{n-t}\cdot\frac{1}{1-f} + \epsilon\right) f\frac{|S|}{\ln 2} \leq \left(1-\frac{2\delta}{3}\right) f\frac{|S|}{\ln 2}\,. \end{aligned}$$

The first relation follows from Definition 18 Eq. 3, the second from the independence of Z_r, the fourth from the bound in Lemma 1 Eq. 5, the fifth and the last from the bounds in [32, Section 13.2.2].

Proposition 2. $Z(S) < \frac{t}{n-t}\cdot\frac{f}{1-f}\cdot\frac{|S|}{\ln 2} + \epsilon f\frac{|S|}{\ln 2} < \left(1+\frac{\delta}{2}\right)\cdot\frac{t}{n-t}X(S) + \frac{\epsilon f|S|}{\ln 2}$. The first relation follows from part (1) of this proof, and the second from the bound in [10, Lemma 11(c)].

Proposition 3. $Y(S) > (1-\epsilon)\,\mathbb{E}[Y(S)] > \left(1-\frac{\delta}{3}\right) f\frac{|S|}{\ln 2} > \left(1-\frac{2\delta}{3}\right) f\frac{|S|}{\ln 2} > Z(S)$. The first inequality follows from Definition 18 Eq. 2, the second from the bound in Lemma 1 Eq. 4, and the last one from part (1) of this proof. □

Theorem 4 (Entropic Growth). *Typical executions of PoEM satisfy the Entropic Growth property with $s = \lambda$ and $\tau = (1-\epsilon)\frac{f}{\ln 2}$.*

Proof. Consider a typical PoEM execution in which an honest party has a chain C_1 at round r_1 and adopts a chain C_2 at round $r_2 \geq r_1 + s$. Let $S = \{r_1, \ldots, r_2 - 1\}$. Then $|S| \geq s = \lambda$ and, applying Definition 18 we obtain $X(S) > (1-\epsilon)\,\mathbb{E}[\underline{X}(S)]$. By Lemma 1, $\mathbb{E}[\underline{X}(S)] \geq \frac{f}{\ln 2}|S|$. Hence, $X(S) > (1-\epsilon)\frac{f}{\ln 2}|S|$. By Lemma 7, $\overline{\overline{\text{WORK}}}(C_2) \geq \overline{\overline{\text{WORK}}}(C_1) + X(S)$, as desired. □

Lemma 9 (Entropic Patience). *In a typical execution, any chained real work $k \geq 2\lambda\frac{f}{1-f}\frac{1}{\ln 2} + 8$ is computed in more than $\frac{k-8}{2\frac{f}{1-f}\frac{1}{\ln 2}} \geq \lambda$ consecutive rounds.*

Proof. Assume, towards a contradiction, there is a set of consecutive rounds S' in which the chained real work k was computed and $|S'| \leq \frac{k-8}{2\frac{f}{1-f}\frac{1}{\ln 2}}$. It holds that $X(S') + Z(S') \geq k$. Then, there is a set $S \supseteq S'$ of consecutive rounds with $|S| = \left\lceil \frac{k-8}{2\frac{f}{1-f}\frac{1}{\ln 2}} \right\rceil + 1 < \frac{k-8}{2\frac{f}{1-f}\frac{1}{\ln 2}} + 2$ such that $X(S) + Z(S) \geq X(S') + Z(S') \geq k$. However, because $|S| > \lambda$, typicality applies and from Lemma 8 we obtain $X(S) < (1+\epsilon)\,\mathbb{E}[\overline{X}(S)] \leq (1+\epsilon)\,\mathbb{E}[\overline{X}_r]|S| < (1+\epsilon)\frac{f}{1-f}\frac{|S|}{\ln 2}$ and $Z(S) < (1+$

$\epsilon)\,\mathbb{E}[Z(S)] \leq (1+\epsilon)\,\mathbb{E}[Z_r]|S| < (1+\epsilon)\frac{t}{n-t}\frac{f}{1-f}\frac{|S|}{\ln 2} < (1+\epsilon)(1-\delta)\frac{f}{1-f}\frac{|S|}{\ln 2}$. Hence,

$$X(S) + Z(S) < (1+\epsilon)\frac{f}{1-f}\frac{|S|}{\ln 2}(1+1-\delta) < 2\frac{f}{1-f}\frac{|S|}{\ln 2} < 2\frac{k-8}{2\frac{f}{1-f}\frac{1}{\ln 2}}\frac{f}{1-f}\frac{1}{\ln 2} + 4\frac{f}{1-f}\frac{1}{\ln 2} = k - 8 + 4\frac{f}{1-f}\frac{1}{\ln 2} = k - 4\left(2 - \frac{f}{1-f}\frac{1}{\ln 2}\right) < k\,.$$

The second inequality follows from the fact that $\epsilon = \frac{\delta}{6} \Rightarrow (1+\epsilon)(2-\delta) < 2$. The last inequality follows from $f < \frac{1}{2} \Rightarrow 2 - \frac{f}{1-f}\frac{1}{\ln 2} < 0$. □

Corollary 4. *In a typical execution of PoEM, for any honest party P and any round r it holds that $\overline{\overline{\text{WORK}}}([:-k]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r) < 2k$.*

Proof. From Entropic Patience (Lemma 9), every block has less than k real work. Therefore, $\overline{\overline{\text{WORK}}}([-k]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r) < k$. From the definition of the slicing notation ($[:]\overline{\overline{\lhd}}$) it holds that $\overline{\overline{\text{WORK}}}(([-k:]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r)[1:]) \leq k$. Summing the two constituents, we obtain

$$\overline{\overline{\text{WORK}}}([-k:]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r) = \overline{\overline{\text{WORK}}}(([-k:]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r)[1:]) + \overline{\overline{\text{WORK}}}([-k]\overline{\overline{\lhd}}^{P}\boldsymbol{C}_r) < 2k\,.$$

□

Lemma 10 (Entropic Common Prefix Lemma). *For all rounds r, and all honest parties P_1, P_2, where P_1 has C_1 and P_2 adopts C_2 at round r of a typical PoEM execution, it holds that $[:-k]\overline{\overline{\lhd}}C_1 \preccurlyeq C_2$ and $[:-k]\overline{\overline{\lhd}}C_2 \preccurlyeq C_1$ for $k = 2\lambda\frac{f}{1-f}\frac{1}{\ln 2} + 8$.*

Proof. Consider an execution as in the statement and suppose, towards a contradiction, that $[:-k]\overline{\overline{\lhd}}C_1 \not\preccurlyeq C_2$ or $[:-k]\overline{\overline{\lhd}}C_2 \not\preccurlyeq C_1$. Consider the last block B^* with index i^* on the common prefix of C_1 and C_2 that was computed by an honest party and let r^* be the round at which it was computed; if no such block exists let $r^* = 0$. Define the set of rounds $S = \{i : r^* < i < r\}$. We claim that $Z(S) \geq Y(S)$.

We show this by pairing all real work of blocks computed by honest parties during convergence opportunities in S with adversarial real work computed during S. Let $\mathcal{Y}(S)$ be the set of honestly produced blocks in convergence opportunities during S, and $\Xi = \bigcup\{I(B) : B \in \mathcal{Y}(S)\}$.

Note that, if $\Xi \neq \emptyset$, then $\inf \Xi \geq \max I(B^*)$ because the chain ending in block B^* was diffused at round r^*, and all honestly produced blocks after round r^* are extending a chain with greater or equal real work. Also note that $\overline{\overline{\text{WORK}}}(C_1) \geq \max \Xi$ and $\overline{\overline{\text{WORK}}}(C_2) \geq \max \Xi$ because the honest party that computed the chain with work $\max \Xi$ diffused it and any chain adopted by honest parties at any later round should have at least $\max \Xi$ work. Hence, for every $\xi \in \Xi$ it holds that $[\xi]\overline{\overline{\lhd}}C_1 \neq \bot$ and $[\xi]\overline{\overline{\lhd}}C_2 \neq \bot$.

We now argue that for every $\xi \in \Xi$ either block $[\xi]\overline{\overline{\lhd}}C_1$ or block $[\xi]\overline{\overline{\lhd}}C_2$ is adversarial. If the block lies on the common prefix of C_1 and C_2, namely $[\xi]\overline{\overline{\lhd}}C_1 = [\xi]\overline{\overline{\lhd}}C_2$, then it is adversarial by the definition of B^*. Otherwise, there is one block in C_1 and another one in C_2, and by Lemma 6, it holds that $[\xi]\overline{\overline{\lhd}}C_1$ and $[\xi]\overline{\overline{\lhd}}C_2$ cannot both be honest. This completes the proof of the claim $Z(S) \geq Y(S)$.

All the chained real work $\max(\overline{\overline{\text{WORK}}}(C_1[i^*{:}]), \overline{\overline{\text{WORK}}}(C_2[i^*{:}])) \geq k$ was produced during $S \cup \{r^*\}$. Hence, from Lemma 9, $|S \cup \{r^*\}| > \lambda \Rightarrow |S| \geq \lambda$ and the properties of a typical execution apply. Therefore, by Lemma 8, $Z(S) < Y(S)$ which contradicts the previous claim. □

Theorem 5 (Entropic Common Prefix). *Typical executions of PoEM satisfy Entropic Common Prefix with* $k = 2\lambda\frac{f}{1-f}\frac{1}{\ln 2} + 8$.

Proof. Consider a typical execution and suppose, towards a contradiction, that Common Prefix is violated, and let r_2 be the first round during which it is violated. At r_2 there is an honest party P_2 who adopts chain C_2 inconsistent with the chain C_1 adopted by an honest party P_1 at a round $r_1 \leq r_2$, namely $[{:}-k]\overline{\overline{\lhd}}C_1 \not\preceq C_2$.

Case $r_1 < r_2$**.** At round r_2, party P_1 has a chain C, which it adopted at $r_2 - 1$ (not excluding the case where $C = C_1$). It holds that $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq C$ due to the minimality of r_2 (otherwise, the Common Prefix virtue would have been broken at $r_2 - 1$ by chains C_1 and C). Furthermore, $\overline{\overline{\text{WORK}}}(C) \geq \overline{\overline{\text{WORK}}}(C_1)$ due to the heaviest chain rule followed by P_1. Therefore, $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq [{:}-k]\overline{\overline{\lhd}}C$. By the Common Prefix lemma, we have $[{:}-k]\overline{\overline{\lhd}}C \preceq C_2$ (at r_2, party P_1 has C and party P_2 adopts C_2). By transitivity of $\preceq$, we have $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq C_2$, which contradicts the violation of Common Prefix.

Case $r_1 = r_2$**.** Let C be the chain that P_1 adopts at $r_1 + 1$ (not excluding the case where $C = C_1$). By the Common Prefix lemma, we have that $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq C$ (at r_1+1, party P_1 adopts C and has C_1). Furthermore, $\overline{\overline{\text{WORK}}}(C) \geq \overline{\overline{\text{WORK}}}(C_1)$ due to the heaviest chain rule followed by P_1. Because $\overline{\overline{\text{WORK}}}(C) \geq \overline{\overline{\text{WORK}}}(C_1)$, therefore $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq [{:}-k]\overline{\overline{\lhd}}C$. By the Common Prefix lemma, we have that $[{:}-k]\overline{\overline{\lhd}}C \preceq C_2$ (at r_1+1, party P_1 adopts C and party P_2 has C_2). By transitivity of $\preceq$, we have $[{:}-k]\overline{\overline{\lhd}}C_1 \preceq C_2$, which contradicts the violation of Common Prefix. □

Theorem 6 (Entropic Quality). *Typical executions of PoEM satisfy the Entropic Quality property with* $\ell = 2\lambda\frac{f}{1-f}\frac{1}{\ln 2} + 8$ *and* $\mu = 1 - (1 + \frac{\delta}{2})\frac{t}{n-t} - \frac{\epsilon}{1-\epsilon}$.

Proof. Suppose, towards a contradiction, that there is a chain quality violation in a typical PoEM execution. Then there is an honest party P who adopts a chain C_3 at round r for which chain quality is violated. This means there are u, v such that the chain $C_1 = C_3[u{:}v]$ has $\overline{\overline{\text{WORK}}}(C_1) \geq \ell$ and quality lower than μ, namely the sum x of works of all honestly generated blocks in C_1 is less than $\mu\overline{\overline{\text{WORK}}}(C_1)$. Consider the minimum real work chain $C_2 = C_3[u'{:}v']$ such that C_1 is fully included in C_2 (i.e., $u' \leq u$ and $v' \geq v$) with the following properties:

1. $C_3[u']$ was computed by an honest party P_1 (this will exist because $C_3[0]$ is the genesis block, which is honestly generated) at some round r_1 (letting $r_1 = 0$ if $u' = 0$).
2. $C_3[v']$ was the tip of the adopted chain by an honest party P_2 at some round r_2 (this will exist because P adopts C_3).

Let $L = \overline{\overline{\text{WORK}}}(C_2)$ and $S = \{r_1, \ldots, r_2 - 1\}$. Note that, by causality, all the work L was computed in S. By the supposition, we have $x < \mu\ell \leq \mu L$.

We have that $Z(S) \geq L - x$. To see this, observe that, by the minimality of C_2, all the blocks with heights $u', \ldots, u$ as well as the blocks with heights $v, \ldots, v'$ were computed by the adversary, and so the only honest real work computed within L is x.

Additionally, $L \geq X(S)$. To see this, note that at round r_1, party P_1 produced $C_3[u']$, and so every honest party adopts a chain of real work at least $\overline{\overline{\text{WORK}}}(C_3[u':])$ from round $r_1 + 1$ onwards. Therefore, by Lemma 7, at round r_2, every honest party adopts a chain of real work at least $\overline{\overline{\text{WORK}}}(C_3[u':]) + X(S)$. But we know that P_2 adopts a chain of real work $\overline{\overline{\text{WORK}}}(C_3[u':]) + L$, and so $L \geq X(S)$.

Therefore,

$$Z(S) \geq L - x > (1 - \mu)L \geq (1 - \mu)X(S) \geq ((1 + \frac{\delta}{2}) \cdot \frac{t}{n - t} + \frac{\epsilon}{1 - \epsilon})X(S)\,.$$

The last inequality follows from replacing the value of μ from the statement. By Lemma 9, $|S| > \lambda$ and typical bounds apply. Therefore, $X(S) > (1 - \epsilon)\,\mathbb{E}[\underline{X}(S)] = (1 - \epsilon)\frac{f}{\ln 2}$ and, from this and the previous inequality, $Z(S) \geq (1 + \frac{\delta}{2}) \cdot \frac{t}{n-t} X(S) + \epsilon f \frac{|S|}{\ln 2}$. However, this contradicts the bound in Lemma 8. □

Lemma 11 (Confirmation Separation). *For any function $k = k(\kappa)$, it holds that, in a PoEM execution,* $\Pr[\exists C : [:-k]\overline{\overline{\lhd}}C \neq [:-k] \lhd C]$ *is negligible in κ.*

Proof. Consider the event that there exists a *chain* C with $[:-k]\overline{\overline{\lhd}}C \neq [:-k] \lhd C$. Since the prefixes differ, the suffixes must also differ, namely $[-k:]\overline{\overline{\lhd}}C \neq [-k:] \lhd C$. Let $C^* = [-k:] \lhd C$. By the slicing notation, it directly follows that $\text{WORK}(C^*) \geq k$. Additionally, we observe that $k > \overline{\overline{\text{WORK}}}(C^*)$ (otherwise, $[-k:]\overline{\overline{\lhd}}C = [-k:] \lhd C$). Define BORDERLINE the bad event that there exists some subchain C^* in the execution such that $\text{WORK}(C^*) \geq k > \overline{\overline{\text{WORK}}}(C^*)$. We have shown that $\exists C : [:-k]\overline{\overline{\lhd}}C \neq [:-k] \lhd C \to$ BORDERLINE.

It suffices to show that $\Pr[\text{BORDERLINE}]$ is negligible.

Let E_1 be the event that there exists a subchain C^* such that $\text{WORK}(C^*) - \overline{\overline{\text{WORK}}}(C^*) \geq Lqn2^{\kappa/2}$, and let $p_1 = \Pr[E_1]$. Let E_2 be the event that there exists a subchain C^* such that $k - Lqn2^{\kappa/2} \leq \overline{\overline{\text{WORK}}}(C^*) < k$, and let $p_2 = \Pr[E_2]$. We observe that $\neg E_1 \wedge \neg E_2 \to \neg$BORDERLINE, hence BORDERLINE $\to E_1 \vee E_2$. Therefore, from the union bound, we have

$$\Pr[E_1] + \Pr[E_2] = p_1 + p_2 \geq \Pr[E_1 \vee E_2] \geq \Pr[\text{BORDERLINE}].$$

From Corollary 2, we have that p_1 is negligible.

Let us calculate the probability p_2. Consider all the, at most $(nqL)^2$, subchains $C_1^*, C_2^*, \ldots, C_{(nqL)^2}^*$ appearing in an execution, and consider an aribtrary subchain C_i^* among them. Because the work of each block B in C_i^* is distributed as $\mathsf{Exp}(\ln 2)$, the points $(\overline{\overline{\text{WORK}}}(C_i^*[:1]), \overline{\overline{\text{WORK}}}(C_i^*[:2]), \ldots, \overline{\overline{\text{WORK}}}(C_i^*))$ correspond to the initial $\overline{\overline{\text{WORK}}}(C_i^*)$ segment of a Poisson stochastic process with rate $\ln 2$ (do not confuse the *work* between blocks, which exactly follows a Poisson process with rate $\ln 2$, and the *time* between consecutive block generation). We are interested in the number of Poisson points that fall within the interval $[k - Lqn2^{-\kappa/2}, k)$. This is a random variable distributed as a Poisson distribution with rate $Lqn2^{-\kappa/2}\lambda$. Let F_i be the bad event that the number of points of the process corresponding to the subchain C_i^* that fall within the interval $[k-Lqn2^{-\kappa/2}, k)$ is at least one. Using the probability mass function of the Poisson distribution evaluated at 0 we have $\Pr[F_i] = 1-\Pr[\neg F_i] = 1-e^{-\lambda Lqn2^{-\kappa/2}} = 1 - 2^{-(\lg e)\lambda Lqn2^{-\kappa/2}} < (\lg e)\lambda Lqn2^{-\kappa/2}$, where the last inequality follows from Lemma 12. Taking a union bound over all subchains, we have $p_2 = \Pr[E_2] = \Pr[\bigcup_{i=1}^{(nqL)^2} F_i] \leq \sum_{i=1}^{(nqL)^2} \Pr[F_i] = (nqL)^2 \Pr[F_i] < (nqL)^3(\lg e)\lambda 2^{-\kappa/2}$ which is negligible. Therefore, $\Pr[\textsc{Borderline}]$ is negligible. □

Theorem 1 (PoEM is Safe). *Typical executions of PoEM are safe.*

Proof. Consider any two honest parties P_1, P_2 and any rounds r_1, r_2. Let C_1, C_2 be the chains that P_1, P_2 adopt at rounds r_1, r_2 respectively. From Entropic Common Prefix (Theorem 5), it follows that if $r_1 \leq r_2$, then $[:-k]\overline{\overline{\lhd}}C_1 \preccurlyeq C_2$; and if $r_2 \leq r_1$, then $[:-k]\overline{\overline{\lhd}}C_2 \preccurlyeq C_1$. In both cases, it follows that $[:-k]\overline{\overline{\lhd}}C_1 \sim [:-k]\overline{\overline{\lhd}}C_2$. From Confirmation Separation (Lemma 11), it follows that $[:-k] \lhd C_1 \sim [:-k] \lhd C_2$. Therefore, for the ledgers ${}^{P_1}\boldsymbol{L}_{r_1}, {}^{P_2}\boldsymbol{L}_{r_2}$ returned when READ is invoked on parties P_1, P_2 after rounds r_1, r_2 respectively, it holds that ${}^{P_1}\boldsymbol{L}_{r_1} \sim {}^{P_2}\boldsymbol{L}_{r_2}$.

Theorem 2 (PoEM is Live). *Typical executions of PoEM are live with parameter* $u = \max(\left\lceil \frac{\ell+2k}{(1-\epsilon)\tilde{f}} \ln 2 \right\rceil, s)$.

Proof. Consider any round r. Because $u \geq s$, invoking Entropic Growth(s, τ) (Theorem 4), we conclude that for all honest parties P and all rounds $r' \geq r+u$, it holds that $\overline{\overline{\text{WORK}}}({}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]) \geq u\tau \geq \ell + 2k$. From Corollary 4, it follows that $\overline{\overline{\overline{\text{WORK}}}}([:-k]\overline{\overline{\lhd}}\,{}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]) \geq \ell$. Invoking Entropic Quality (Theorem 6), it holds that in the chain segment $[:-k]\overline{\overline{\lhd}}\,{}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]$, there is at least one honestly generated block that was produced after round r.

Now, consider that an honest party attempts to inject a transaction tx at round r. At the beginning of round $r+1$, all honest parties receive tx and include it in their mempool [32, Section 5.7]. Hence, all honest blocks produced after round r will either include, or extend a chain that includes transaction tx. Because of this and the above, for all honest parties P and rounds $r' \geq r+u$, we conclude that $\mathsf{tx} \in {}^{P}\boldsymbol{L}_{r'}$. □

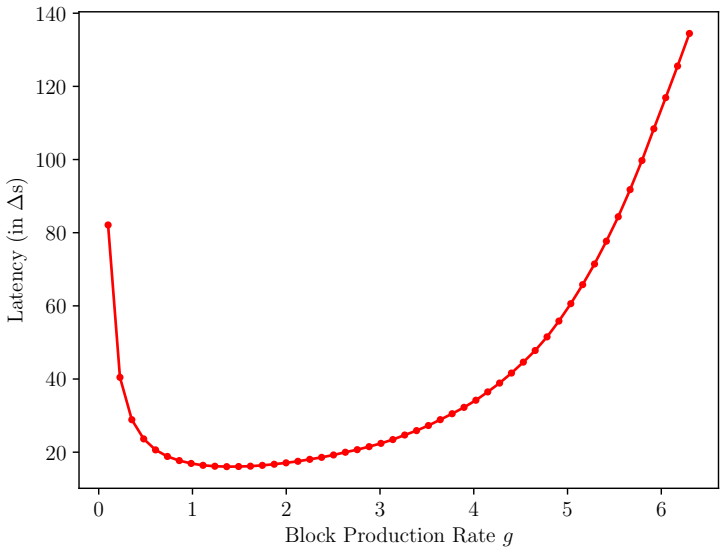


Fig. 4: Fixing $\beta = 0.2$ and $\gamma = 0$, we plot the latency of PoEM parameterized under different block production rates g. The results were obtained by running 100,000 Monte Carlo iterations per point.

Corollary 1 (PoEM is Secure). *PoEM is secure with overwhelming probability.*

Proof. Executions are typical with overwhelming probability (Theorem 3). Typical executions are safe (Theorem 1), and live (Theorem 2), from which security follows. □

B Further Experimental Results

B.1 The Block Production Rate

As expected, the confirmation latency of PoEM behaves convexly as the block production rate g is varied. This is illustrated in Figure 4 for fixed values of β and γ. The simulation methodology is the same as in Section 4. For small block production rates, close to zero, we get high latency because blocks are produced very slowly. For large block production rates, we get high latency because honest blocks effect many forks, whereas the adversarial blocks are all chained in series, and a large confirmation parameter k is required for safety. We get optimal latency somewhere in the middle.

B.2 The Bias Parameter

In our analysis, we assumed $\gamma = 0$ for simplicity, but the γ parameter seems to be a promising knob to tune the performance of PoEM. While we leave the analytic

treatment of the optimal γ for future work, we note here that, for given values of g and β, the delay of the system behaves as illustrated in Figure 5 for varying values of γ. It is convex, and for $\gamma \to \infty$, the delay asymptotically approaches a value smaller than Bitcoin's. Indeed, if γ is made sufficiently large, the bias parameter dominates against the $-\lg \frac{H(B)}{T}$ term, and the system behaves as if each block counts the same. However, even with $\gamma \to \infty$, PoEM provides a tie-breaker which helps with latency. In this regime, this tie-breaker is as good as any.

C Discussion & Future Work

Composability. Because PoEM changes only the proof-of-work inequality, it can be composed with other previously proposed improvements upon PoW to give cumulative benefits. Such examples are different block topologies like Bitcoin-NG [7], FruitChains [25], Prism [1], Parallel Chains [8], PHANTOM [29], SPECTRE [26], GhostDAG [27], GHOST [28], Ledger Combiners [9] and HLCR [13]. A formal proof of the composability of PoEM with these protocols is left for future work.

Bias. Whereas our security analysis was conducted for $\gamma = 0$, the real-world PoEM deployment uses positive values for γ. We also used (among others) positive values for γ when conducting our experiments in Section 4. We have experimentally observed that increasing γ improves the rate at which the sum of independent random variables each distributed as $\mathsf{Bern}(\cdot)(\gamma + \mathsf{Exp}(\frac{1}{\ln 2}))$ converges, but the expectations deteriorate as far as security is concerned. We suspect that, for a given acceptable probability of failure given by the security parameter κ, there is an optimal configuration (g, γ) (and a corresponding k) that minimizes the confirmation delay. This is supported by our experimental evidence in Section B. Can this optimal configuration be found analytically?

Additionally, we know that, at the operating limit of $\gamma \to \infty$, the protocol is exactly Bitcoin with an arbitrary tie-breaker, so we have proofs of security for both $\gamma = 0$ and $\gamma \to \infty$, but not for $0 < \gamma < \infty$. We leave the full formal analysis of these questions for future work, although we note that the relevant Chernoff bounds for non-negative biases γ are well-behaved, as we analytically prove in Lemma 13.

Work functions. We used the function $\overline{\overline{\mathrm{WORK}}}(B) = \gamma - \lg \frac{\overline{\overline{H}}(B)}{T}$. This definition corresponds to the intuitive idea that each successful query to the random oracle reduces the number of possible evolutionary paths of the system, thus reducing its "entropy" (hence the name *PoEM*, Proof of Entropy Minima). An open question is whether this function is optimal, or whether a different function optimizes confirmation latency.

Tight bounds. In our proofs, we have used the conservative configuration $f = \frac{\delta}{6}$, which yields a small value for the honest block production rate g, following the model of Backbone [10]. Follow up works in Bitcoin [5,12] have shown tighter operating limits for Bitcoin, and we expect that similar results can be obtained for PoEM. We have experimentally demonstrated that consensus is achieved

with higher values of g when the honest parties play against a private mining adversary. This instills confidence, because we know from the work in [5] that this private mining attack is indeed the best possible attack against Bitcoin in the continuous-time domain [14]. However, no such result exists for PoEM. Showing that PoEM is secure for high values of g against *any* adversary, or that indeed the private mining attack is also the best possible attack against PoEM, is left for future work. Such an analysis poses technical challenges because, even though PoEM might have better behavior of expected values, the concentration of the random variables is worse than in Bitcoin.

Difficulty adjustment. In our security proof, we assumed a static population in accordance with the Bitcoin Backbone analysis [10]. The practical deployments of both Bitcoin and PoEM adjust their difficulty in response to changes in the miner population. Bitcoin was proven secure in this variable difficulty setting [11]. One of the technical challenges in this analysis is to also consider whether the value γ should be adjusted in response to changes in the miner population. We leave the variable difficulty analysis of PoEM for future work.

DAGs. Some engineering work in our real-world deployment has indicated that using PoEM's fork choice rule in a DAG-based blockchain with a particular topology may be beneficial. More research is needed to explore this direction.

D Mathematical Background

Lemma 12. *For all $0 < y < 1$, it holds that $2^y - 1 < y$ and $1 - 2^{-y} < y$.*

Proof. For the first part, it suffices to show that $(y+1)^{1/y} > 2$, as this implies that $2^y < y + 1$ and, ultimately, $2^y - 1 < y$. The inequality $(y+1)^{1/y} > 2$ holds due to Bernoulli's inequality ($(1+x)^r > 1 + rx$ for all $x > 0$ and $r > 1$), when setting $x = y$ and $r = \frac{1}{y}$.

For the second part, it suffices to show that $(1-y)^{1/y} < \frac{1}{2}$. Let $f(y) = (1-y)^{1/y}$ and

$$\begin{aligned}\frac{d}{dy}f(y) = \frac{d}{dy}(1-y)^{1/y} = \frac{d}{dy}e^{\frac{1}{y}\ln(1-y)} = \\ (1-y)^{1/y}\left(-\frac{1}{y(1-y)} - \frac{\ln(1-y)}{y^2}\right) = \\ (1-y)^{1/y}\left(\frac{y-(y-1)\ln(1-y)}{y^2(y-1)}\right)\end{aligned}$$

Letting $\phi(y) = y - (y-1)\ln(1-y)$, we have $\frac{d}{dy}f(y) = (1-y)^{1/y}\left(\frac{\phi(y)}{y^2(y-1)}\right)$. Observe that $\phi(y)$ is continuous and differentiable for $y \in (0,1)$. It holds that

$$\begin{aligned}\frac{d}{dy}\phi(y) = \frac{d}{dy}(y-(y-1)\ln(1-y)) = \\ \frac{d}{dy}(y - y\ln(1-y) + \ln(1-y)) = \\ 1 - \ln(1-y) + \frac{y}{1-y} - \frac{1}{1-y} = \\ 1 - \ln(1-y) - \frac{1-y}{1-y} = -\ln(1-y)\end{aligned}$$

Hence, for $y \in (0,1)$, it holds that $\frac{d}{dy}\phi(y) > 0$ and $\phi(y)$ is increasing. Because $\phi(0) = 0$, it holds that $\phi(y) > 0$ for $y \in (0,1)$. Therefore, for $y \in (0,1)$, it holds that $\frac{d}{dy}f(y) < 0$ and $f(y)$ is decreasing.

Setting $\omega = -\frac{1}{y}$, we have

$$\begin{aligned}\lim_{y\to 0} f(y) = \lim_{y\to 0}(1-y)^{\frac{1}{y}} = \lim_{\omega\to-\infty}\left(1+\frac{1}{\omega}\right)^{-\omega} = \\ \frac{1}{\lim_{\omega\to-\infty}\left(1+\frac{1}{\omega}\right)^{\omega}} = \frac{1}{e}\end{aligned}$$

Pick an arbitrary $0 < \epsilon < \frac{1}{2} - \frac{1}{e}$. By the definition of the limit, there exists a $\delta > 0$ such that for all $y \in (0,\delta)$, it holds that $|f(y) - \frac{1}{e}| < \epsilon \Leftrightarrow f(y) < \frac{1}{2}$. Hence, for $y \in (0,1)$, because $f(y)$ is continuous and decreasing, it holds that $f(y) < \frac{1}{2}$. □

Lemma 13 (Concentration of Bern × Exp). *Let $\{A_i\}_{i\in[n]}$ and $\{B_i\}_{i\in[n]}$ be two families of i.i.d. random variables, all mutually independent, with A_i distributed as $\mathsf{Bern}(p)$ and B_i distributed as $\gamma + \mathsf{Exp}(\lambda)$, the exponential distribution shifted by a constant $\gamma \geq 0$. Let $X_i = A_iB_i$, and $X = \sum_{i=1}^{n} X_i$. Then for any $0 < \epsilon < 1$, it holds that $\Pr[X > (1+\epsilon)\,\mathbb{E}[X]] < e^{-\Omega(n)}$ and $\Pr[X < (1-\epsilon)\,\mathbb{E}[X]] < e^{-\Omega(n)}$, which is negligible in n.*

Proof. $\mathbb{E}[X_i] = \mathbb{E}[A_iB_i] = \mathbb{E}[A_i]\,\mathbb{E}[B_i] = p(\gamma + \frac{1}{\lambda})$, therefore $\mathbb{E}[X] = np(\gamma + \frac{1}{\lambda})$. For the moment generating functions we have

$$\begin{aligned}\mathbb{E}[e^{tX_i}] = \mathbb{E}[e^{tA_iB_i}] = \\ \mathbb{E}[e^{tA_iB_i}|A_i = 0]\Pr[A_i = 0] \\ + \mathbb{E}[e^{tA_iB_i}|A_i = 1]\Pr[A_i = 1] = \\ \mathbb{E}[e^{tA_iB_i}|A_i = 0](1-p) + \mathbb{E}[e^{tA_iB_i}|A_i = 1]p = \\ (1-p) + p\,\mathbb{E}[e^{tB_i}] = (1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\,.\end{aligned}$$

$$\mathbb{E}[e^{tX}] = \mathbb{E}[e^{t\sum_{i=1}^{n} X_i}] = \mathbb{E}[\prod_{i=1}^{n} e^{tX_i}] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}] =$$
$$\mathbb{E}[e^{tA_iB_i}]^n = \left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right]^n = e^{n\ln\left[(1-p)+pe^{t\gamma}\frac{\lambda}{\lambda-t}\right]}.$$

For all $0 < t < \lambda$:

$$\Pr[X > (1+\epsilon)\,\mathbb{E}[X]] = \Pr[X > (1+\epsilon)np(\gamma + \frac{1}{\lambda})] = \Pr[e^{tX} > e^{t(1+\epsilon)np(\gamma+\frac{1}{\lambda})}]$$
$$\leq \mathbb{E}[e^{tX}]e^{-t(1+\epsilon)np(\gamma+\frac{1}{\lambda})} = e^{n\ln\left[(1-p)+pe^{t\gamma}\frac{\lambda}{\lambda-t}\right]-nt(1+\epsilon)p(\gamma+\frac{1}{\lambda})}.$$

Consider the factor $f(t) = \ln\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda-t}\right] - t(1+\epsilon)p(\gamma + \frac{1}{\lambda})$ in front of n in the exponent. Taking its derivative with respect to t:

$$\frac{d}{dt}\left(\ln\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right] - t(1+\epsilon)p\left(\gamma + \frac{1}{\lambda}\right)\right) =$$
$$\frac{1}{(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda-t}}\frac{d}{dt}\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right] - (1+\epsilon)p\left(\gamma + \frac{1}{\lambda}\right) =$$
$$\frac{pe^{t\gamma}\frac{\lambda}{\lambda-t}(\gamma + \frac{1}{\lambda-t})}{(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda-t}} - (1+\epsilon)p\left(\gamma + \frac{1}{\lambda}\right)$$

At $t = 0$ we have $f(0) = 0$ and

$$\frac{d}{dt}f(0) = \frac{p(\gamma + \frac{1}{\lambda})}{1-p+p} - (1+\epsilon)p\left(\gamma + \frac{1}{\lambda}\right) =$$
$$p\left(\gamma + \frac{1}{\lambda}\right) - (1+\epsilon)p\left(\gamma + \frac{1}{\lambda}\right) = -\epsilon p\left(\gamma + \frac{1}{\lambda}\right) < 0\,.$$

Since $\frac{d}{dt}f$ is continuous at 0 and $\frac{d}{dt}f(0) < 0$, there must exist some $0 < t^* < \lambda$ such that for all $0 < t < t^*$ it holds that $\frac{d}{dt}f(t) < 0$. Because f is continuous and differentiable in $[0, t^*]$, by the Mean Value Theorem, there must exist some $\xi \in (0, t^*)$ such that $\frac{d}{dt}f(\xi) = \frac{f(t^*)-f(0)}{t^*-0} = \frac{f(t^*)}{t^*}$. Since $t^* > 0$ and $\frac{d}{dt}f(\xi) < 0$, therefore $f(t^*) < 0$. This t^* makes the factor in front of n in the exponent negative, and therefore gives us a bound for which $\Pr[X > (1+\epsilon)\,\mathbb{E}[X]] < e^{-\Omega(n)}$.

For all $t < 0$:

$$\Pr[X < (1-\epsilon)\,\mathbb{E}[X]] = \Pr\left[X < (1-\epsilon)np(\gamma + \frac{1}{\lambda})\right] = \Pr[e^{tX} > e^{t(1-\epsilon)np(\gamma+\frac{1}{\lambda})}]$$
$$\leq \mathbb{E}[e^{tX}]e^{-t(1-\epsilon)np(\gamma+\frac{1}{\lambda})}$$
$$= e^{n\ln\left[(1-p)+pe^{t\gamma}\frac{\lambda}{\lambda-t}\right]-t(1-\epsilon)np(\gamma+\frac{1}{\lambda})}$$

Consider the factor $f(t) = \ln\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right] - t(1-\epsilon)p(\gamma + \frac{1}{\lambda})$ in front of n in the exponent. Taking its derivative with respect to t:

$$\frac{d}{dt}\left(\ln\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right] - t(1-\epsilon)p(\gamma + \frac{1}{\lambda})\right) =$$
$$\frac{1}{(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}}\frac{d}{dt}\left[(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}\right] - (1-\epsilon)p(\gamma + \frac{1}{\lambda}) =$$
$$\frac{pe^{t\gamma}\frac{\lambda}{\lambda - t}(\gamma + \frac{1}{\lambda - t})}{(1-p) + pe^{t\gamma}\frac{\lambda}{\lambda - t}} - (1-\epsilon)p(\gamma + \frac{1}{\lambda})$$

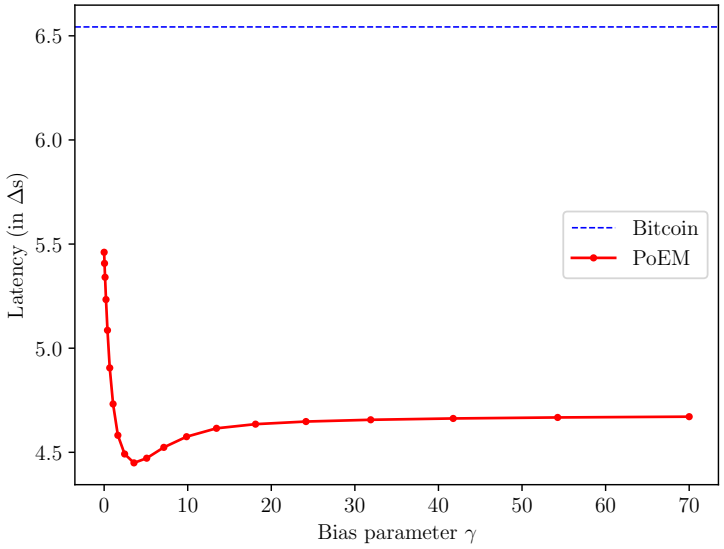
At $t = 0$ we have $f(0) = 0$ and

$$\frac{d}{dt}f(0) = \frac{p(\gamma + \frac{1}{\lambda})}{(1-p) + p} - (1-\epsilon)p(\gamma + \frac{1}{\lambda}) =$$
$$p(\gamma + \frac{1}{\lambda}) - (1-\epsilon)p(\gamma + \frac{1}{\lambda}) = \epsilon p(\gamma + \frac{1}{\lambda}) > 0\,.$$

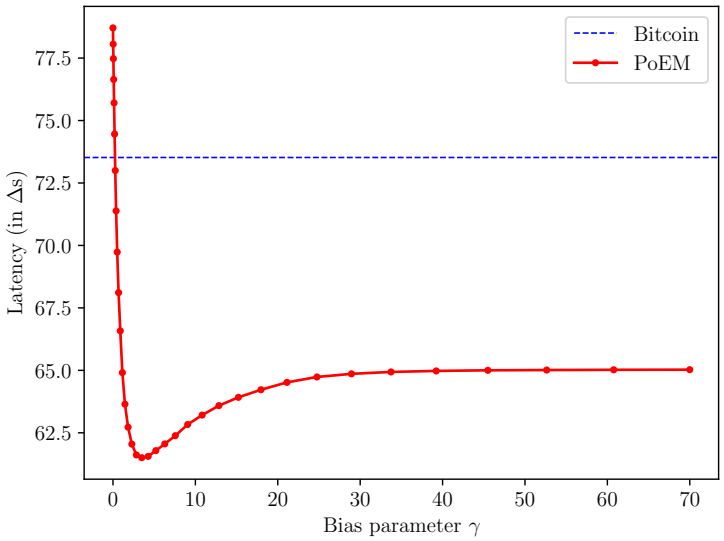
Since $\frac{d}{dt}f$ is continuous at 0 and $\frac{d}{dt}f(0) > 0$, there must exist some $t^* < 0$ such that for all $t^* < t < 0$ it holds that $\frac{d}{dt}f(t) > 0$. Because f is continuous and differentiable in $[t^*, 0]$, by the Mean Value Theorem, there must exist some $\xi \in (t^*, 0)$ such that $\frac{d}{dt}f(\xi) = \frac{f(0) - f(t^*)}{0 - t^*} = \frac{f(t^*)}{t^*}$. Since $t^* < 0$ and $\frac{d}{dt}f(\xi) > 0$, therefore $f(t^*) < 0$. This t^* makes the factor in front of n in the exponent negative, and therefore gives us a bound for which $\Pr[X < (1-\epsilon)\,\mathbb{E}[X]] < e^{-\Omega(n)}$. □

E Acknowledgements

AT and DZ wish to thank Petros Aggelatos for discovering a technical error in the proof of the Block Approximation Lemma; Odysseas Sofikitis for proof reading and pinpointing a few off-by-one errors; Giulia Scaffino for reading early drafts of this work and providing valuable feedback; Alexandra Stavrianidi for her mathematical insights on probabilities and statistics; and David Tse for clarifying technical pitfalls in our Everything-is-a-Race–style simulations.



(a) $\beta = 0.1$ and $g = 1.7$



(b) $\beta = 0.3$ and $g = 0.4$

Fig. 5: Fixing β and g, we compare Bitcoin's latency against PoEM's parameterized under different γ values (denser around the points of interest). We observe the plot is convex and for large γ, the latency approaches a value lower than Bitcoin's. The results were obtained by running 100,000 simulations per point.