A Better Proof-of-Work Fork Choice Rule

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

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

Jul 1, 2023
Last update: February 6, 2024

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 decrease the confirmations required, yielding a 28.5% improvement in *confirmation delay* or, dually, safely increase the block production rate, yielding a 16.3% improvement in *throughput*, as compared to the vanilla Bitcoin proof-of-work fork choice rule. Our modification is at the level of the proof-of-work inequality, and thus can be composed with any other methods to improve latency or throughput that have been proposed in the literature. We report the experimental findings by measuring them on a production-grade implementation of our system, whose testnet is already deployed in the wild. Lastly, we formally prove the security of our new protocol in the Bitcoin Backbone model.

1 Introduction

In Bitcoin [24], a population of *miners* attempt to find *blocks* in the form $B = h \,\|\, x \,\|\, ctr$, where h is a pointer to the previous block, x contains a sequence of *transactions*, and ctr is a *nonce*. The nonce is brute forced by the miner to satisfy the *proof-of-work* inequality $H(B) < T$, where H is a hash function with output in the interval $(0, 1)$ and T is a small *target* in the interval $(0, 1)$. Any B that satisfies this inequality is considered a *valid block*, whereas candidates that do not satisfy the inequality are invalid. Simply put, valid blocks must have hashes that begin with a desired number of 0s. These blocks form linked lists known as *chains*. Among such chains, the *heaviest* chain is chosen as the canonical one.

Some blocks B satisfy the proof-of-work inequality better than others. Namely, they satisfy not only $H(B) < T$, but also $H(B) < \frac{T}{2^w}$ for some $w \in \mathbb{R}^+$. Nevertheless, these heavier blocks are counted all the same when choosing which chain to pick. 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. We build a protocol which retains the same level of security as Bitcoin, while achieving better confirmation latency or transaction throughput, because the number of confirmations can be safely reduced, or the block production rate can be safely increased respectively.

We report on our production implementation of a real-world system in which we employ this new rule, and show that it achieves a 28.5% improvement in confirmation latency and a 16.3% improvement in throughput as compared to Bitcoin. We also theoretically prove our new protocol is secure using the Bitcoin Backbone model. Our theoretical analysis introduces a new technique, the *real-valued random oracle*, which we prove behaves similarly to the usual random oracle. This variant of the random oracle allows for the use of an arsenal of theoretical tools from the continuous domain (such as continuous distributions), easing the theoretical exposition, and may be of independent interest.

Construction overview. The classical Bitcoin protocol uses the heaviest chain fork choice. Among two different chains, the one with the *most work* is chosen as the canonical one, where the work (or difficulty) of a block is defined as $\frac{1}{T}$, with T being the mining target. We modify this rule as follows: Each block counts for its *intrinsic* work $-\lg \frac{H(B)}{T}$, where $H(B)$ denotes the hash of the block target. Intuitively, this corresponds to counting the "number of extra zeroes" at the front of the actually achieved hash of each block, not just accounting for its nominal target. The zeroes at the front of the hash are already guaranteed to be at least $-\lg T$, but can be more. The score of each block is *how many extra zeroes* it has. The chain with the most total intrinsic work is chosen as the canonical chain.

To see the benefits of this rule, first observe that it provides a natural tie-breaking rule: If two honest parties observe the same block tree, they will agree on the canonical chain, regardless of the order of network message arrival. The PoEM protocol outperforms Bitcoin when the block production rate is increased. It is generally desirable to increase the block production rate, because it corresponds to an increase in the chain growth rate, which, in turn, increases the transaction throughput. However, when the expected block interarrival time approaches the network delay, multiple honest blocks can be produced in short succession without allowing for honest nodes to synchronize their views. As a result, multiple honest nodes may produce blocks at the same height, and only one of these will eventually make it to the canonical chain, while the others will be discarded. These discarded blocks do not contribute to the growth of the length of the honest chain. Our protocol is similar to Bitcoin in that, whenever multiple honest blocks are found simultaneously, only one of them survives in the canonical chain, and the rest are discarded. However, the surviving block is the *heaviest* block among them, in terms of intrinsic work, as illustrated in Figure 1. Hence, the discarded work is less. This gives an advantage to the honest parties when they are racing against a private mining attacker, since they can produce chains that grow at a faster rate. This intuition generalizes to arbitrary adversaries. The small change we introduce in the fork choice rule allows us to safely increase the block production rate, as compared to Bitcoin, without sacri-

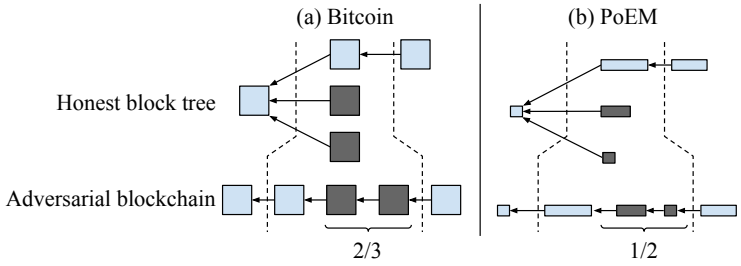


Fig. 1: The same block mining successes awarded to the honest parties (top) or the adversary (bottom) with equal mining power on Bitcoin (left) or PoEM (right) respectively. The adversary can place all blocks in one sequence because she does not incur any network delay. The honest parties, due to the delay, may place blocks at the same height (dashed section of duration Δ). In this example, when 3 honest blocks were found almost simultaneously, 2 out of them were abandoned in Bitcoin and did not make it to the canonical longest chain (top-most chain). In the PoEM example, 2 out of 3 of the honest blocks were abandoned, but the cumulative intrinsic work wasted only happened to be 1/2 of the intrinsic work produced during this interval. We illustrate the intrinsic work of a block by its size.

ficing adversarial resilience. The resulting protocol allows for faster transaction confirmation, or better throughput, while retaining the same level of security.

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. Previous modifications to the fork choice rule include PHANTOM [29], SPECTRE [26], GhostDAG [27], and GHOST [28]. Alternative mechanisms towards improving the latency and throughput of proof-of-work blockchains at the consensus layer 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]. These mechanisms are orthogonal and can be combined with our approach, yielding even further performance gains.

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.

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 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).*

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 κ and orchestrated by an environment $\mathcal{Z}$ which is similar but distinct from 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 $\kappa \in \mathbb{N}$. We assume a synchronous communication network: 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.

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 $\frac{1}{T}$, where T is the nominal target of the block. 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 $-\lg \frac{H(B)}{T}$. 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 work in the following variant of the Random Oracle model [2], in which the random oracle returns a *real number* instead of κ bits of output. One technicality with this model is that the real number cannot be directly returned to the machine invoking the oracle. Instead, we allow the querying machine to choose which bit of the number to obtain.

Definition 5 (Real-Valued Random Oracle). *The* real-valued random oracle H *can be queried with an input value x and a bit index j and returns the value $H(x)[j]$ 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 j-th bit from the binary representation of the real number y. It then remembers the pair (x, y). We denote by $H(x)$ this sampled value y. When queried with x for a subsequent time, it returns the j-th bit of the stored y.*

We denote by $H(x)[i{:}j]$ the query that obtains the slice of $H(x)$ from bit index i to j. We liberally use the rest of the previously introduced slicing notation with the hash output, implying that the random oracle is queried with the desired number of bits. In particular, we will write $\widetilde{H}(x)$ to denote $H(x)[{:}\kappa]$.

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.

```
1: 𝒢
2: C ← [ ]
3: function CONSTRUCTOR(𝒢′)
4:     𝒢 ← 𝒢′                                          ▷ Select Genesis Block
5:     C ← [𝒢]                                         ▷ Add Genesis Block to start of chain
6:     round ← 1
7: end function
8: function EXECUTE_NET(1^κ)
9:     ▷ Receive chains from the network
10:    M̄ ← NET.RECEIVE()
11:    C ← maxvalid(C ∪ M̄)                             ▷ Adopt heaviest chain
12:    x ← INPUT()                                     ▷ Take all transactions in mempool
13:    B ← PoW(x, H̃(C[−1]))                            ▷ Mine a new block
14:    if B ≠ ⊥ then                                   ▷ Successful mining
15:        NET.BROADCAST(C ‖ B)                        ▷ Broadcast mined chain
16:    else
17:        NET.BROADCAST(C)                            ▷ Broadcast adopted chain
18:    end if
19:    round ← round+1
20: end function
21: function READ
22:    return ([:−k] ◁ C).x
23: end function
```

A block is any triplet of the form $B = (h, x, \mathsf{ctr})$, where $h \in \{0,1\}^\kappa$, $x \in \{0,1\}^*$, and $\mathsf{ctr} \in \mathbb{N}$. The hash of block B is denoted as $H(B) = H(h \,\|\, x \,\|\, \mathsf{ctr})$. The h is a reference to a previous block B' and contains the first κ bits of its hash $H(B')[:\kappa]$ (recall that, in the real-valued random oracle model, each hash output has an infinite real-valued bit expansion, but only the first κ bits are included in this reference).

A *chain* C is a sequence of blocks. The chains considered by honest parties begin with a designated block G, the *genesis block*. The genesis block is a constant block known to all parties at the beginning of the execution, which we consider honest by definition.

When a chain C appears in the execution, we say that block $C[j]$ *extends* block $C[i]$ if $i < j$.

Definition 6 (Block Intrinsic Work). *The* intrinsic work *of a block hash A is denoted as* $\textsc{work}(A) = \gamma - \lg A$. *The* approximate intrinsic work *of a block hash A is denoted* $\widetilde{\textsc{work}}(A) = \gamma - \lg A[:\kappa]$, *where* $\gamma \in \mathbb{R}^+$ *is the* bias *parameter of the protocol.*

For genesis, we set $\textsc{work}(G) = 1$ (an arbitrary constant) by convention. To simplify the analysis, we will use set $\gamma = 0$.

In this definition, the full hash value, interpreted as a real number in the interval $(0, 1)$, is used, whereas approximate work uses only the first κ bits of the value. Said differently, for any $A \in (0, 1)$, $\widetilde{\textsc{work}}(A) = -\lg \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa}$.

Definition 7 (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* $\textsc{work}(C) = \sum_{B \in C} \textsc{work}(H(B))$. *The* approximate intrinsic work *of a chain is* $\widetilde{\textsc{work}}(C) = \sum_{B \in C} \widetilde{\textsc{work}}(H(B))$.

Blockchain notation. For chain C, we write $[\alpha] \lhd C$ to denote the i^{th} block of C such that $\textsc{work}(C[:i-1]) < \alpha \leq \textsc{work}(C[:i])$. If $\textsc{work}(C) < \alpha$, then let $[\alpha] \lhd C = \bot$. If α is negative, then $[\alpha] \lhd C$ is defined as the i^{th} block of C such that $\textsc{work}(C[i-1:]) < -\alpha \leq \textsc{work}(C[i:])$. 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. 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 $\Big\|_{B \in C} B.x$.

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 (note that this function is due to 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 H̃(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 10) 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 11, 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 $\widetilde{H}(B) < T$, where the target T is a small real number in the interval $(0, 1)$ satisfying $T = T[:\kappa]$ (this last equality guarantees that T can be stored in κ bits). If a block is found, this block is broadcast[6] to the network using function NET.BROADCAST().

Algorithm 3 The maxvalid algorithm

```
1: function MAXVALID_G(C̄)
2:     C_max ← ϵ
3:     maxwork ← 0
4:     for C ∈ C̄ do
5:         if ¬VALIDATE_G(C) then
6:             continue
7:         end if
8:         thiswork ← W̃ORK(C)                ▷ Computed as Σ_{B∈C} H(B)[:κ]
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
```

We will now analyze the functionality of 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 the provided chains and first checks their validity in Line 9, using function VALIDATE (Algorithm 4). The VALIDATE function remains unchanged compared to the original Bitcoin protocol. The chains that satisfy the validation rule are compared with one another in order to find the chain with the most intrinsic work (hereforth "heaviest chain"). Finally, in Line 14, we return the "best" chain C_{max}.

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

When the time comes to report the stable chain (function READ in Algorithm 1 Line 21), 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 & Deployment

In this section, we report on a real-world implementation and deployment of our protocol, and experimental measurements illustrating the concrete improvements in latency and throughput as compared to vanilla Bitcoin.

4.1 Real-world Deployment

We have implemented and deployed PoEM in a real-world permissionless peer-to-peer setting[7]. The deployment is on a testnet that has been continuously

[7]The source code resides in the following repositories:

1. `https://github.com/gameofpointers/go-quai/tree/poem-sim-honest`
2. `https://github.com/gameofpointers/go-quai/tree/poem-sim-adv`
3. `https://github.com/gameofpointers/go-quai/tree/bitcoin-sim-honest`
4. `https://github.com/gameofpointers/go-quai/tree/poem-sim-adv`
5. `https://github.com/gameofpointers/quai-cpu-miner/tree/sim-miner`

operating for four months. During this period, the network generated over 7.5 million blocks with participation from more than 2000 miners/node operators from the community. Moreover, PoEM has facilitated the confirmation of over 500 million testnet transactions. This deployment maintained an average hash rate exceeding 50GH/s using the ProgPoW [23] hash algorithm. The network participants computed more than 518 petahashes in 4 months. The network operated in a variable difficulty and bias setting with the difficulty ($\frac{2^{256}}{T}$) varying between 18 billion and 790 billion and γ varying between 32 and 38.

4.2 Experimental Methodology

We ran multiple simulations in order to ascertain the optimal configuration for operating both Bitcoin and PoEM in order to achieve the minimum confirmation delay. We simulated an artificial network delay of Δ (measured in seconds) in the communication between the honest parties in order to indirectly control the honest block production rate g, measured in blocks per network delay. We define g to be the number of valid blocks cumulatively produced by the honest parties in one network delay on average, including blocks that were subsequently abandoned (following the terminology in [5]). This corresponds to the average number of successful honest random oracle queries per network delay. In the meantime, we kept the mining target T constant throughout the simulations (instead of varying the network delay Δ and keeping T constant, we could have, equivalently, kept Δ constant and varied the target T). The target T was not dynamically adjusted during the simulation, in order to mimic the static nature corresponding to our theoretical analysis. Our goal was to plot the confirmation delay d (in seconds) as a function of the block production rate g for both systems.

All executions included exactly $n = 49$ parties, of which $t = 12$ were adversarial and $n - t = 37$ were honest. The adversarial ratio was $\beta = \frac{t}{n} = 0.244$. The honest parties ran the honest code of Bitcoin or PoEM respectively. We fixed the adversarial strategy to be the private mining attack, which was proven [5] to be the best possible attack against Bitcoin in the continuous-time domain [14]. This means that the network began with a genesis block given to all honest and adversarial parties. The adversary then mined blocks in private on her own chain, whereas the honest parties mined their own blocktree without intervention by the adversary, following the heaviest chain rule (in Bitcoin) or the most intrinsic work rule (in PoEM) respectively. To emulate the puppetmaster nature of the adversary, we imposed no artificial network delay for the adversary. On the contrary, every honest message was artificially delayed by exactly the maximum delay bound Δ.

Simulations were run on 49 virtual machines in Google Cloud all co-located in the same data center. The configuration of the nodes used was 4 CPU cores, 4 GB of RAM, with a 10 GB SSD running Ubuntu 22.4. The honest party had

6. `https://github.com/gameofpointers/simulation`

an artificial network delay between 100-1000ms (to allow for simulating varying g). This delay was achieved by adding a `sleep` prior to every block message broadcast. The adversary's network delay was only the inherent delay in communication between virtual machines and was <5 ms. The tests were coordinated using Ansible and the machines were given 2 minutes to peer with each other and stabilize at the beginning of each test. Both Bitcoin and PoEM were run on the same codebase, with the exception of the appropriate modification to the proof-of-work inequality.

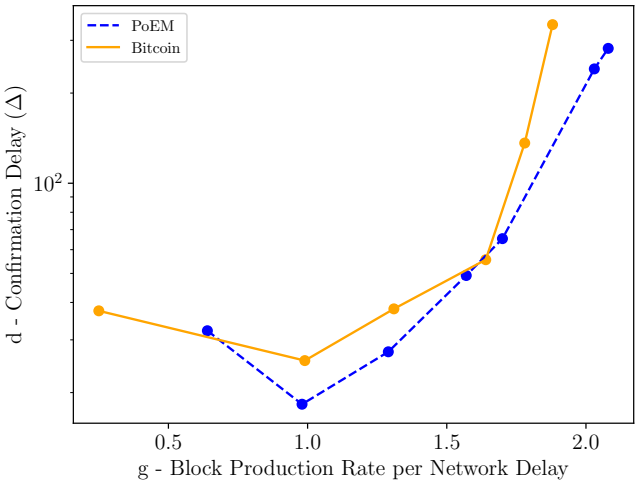
We ran each simulation until one of the honest parties managed to obtain a chain of length 200 blocks, at which point the simulation was halted and its lifetime L (in units of Δ) was measured. We recorded g for that simulation as the number of honest blocks produced divided by L to find the average number of honest blocks produced per network delay (regardless of whether they were ultimately abandoned or not).

For Bitcoin, at every block height $1, \ldots, 200$ that appeared in the simulation, we recorded a flag indicating whether the honest parties or the adversary was ahead. For each configuration of g, we repeated the simulation 100 times in a Monte Carlo fashion, and chose the value k^* to be the block height such that at least 90% of the Monte Carlo simulations indicated that the honest parties were ahead of the adversary from height k^* onwards. This corresponds to accepting a probability of Common Prefix failure of up to 10%. We calculated the average honest chain growth rate f as $f = L/200$ (noting that $f \leq g$ and $0 \leq f \leq 1$, roughly following the terminology of the Analysis section, and observing that, due to the heaviest chain rule, the chain can grow no more than 1 block per network delay when the honest parties are operating alone). The confirmation delay (d) for each configuration g was obtained by dividing this k^* by f to calculate the time needed to get k^* blocks on average.

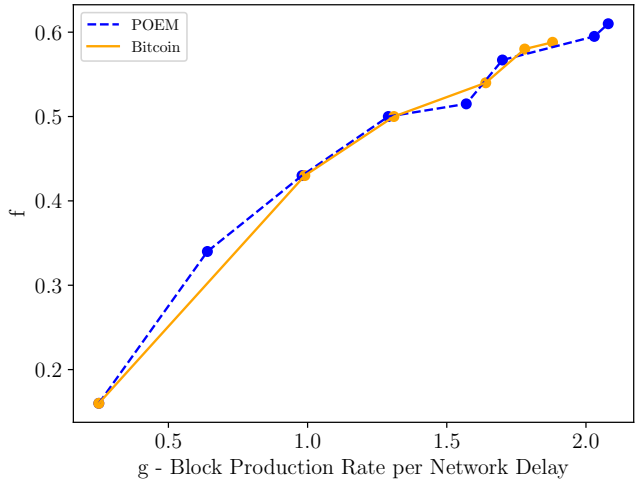
In PoEM's case, at every WORK point in intervals of $\gamma + \frac{1}{\ln 2}$ (corresponding to the expected amount of work per valid block), namely $\gamma + \frac{1}{\ln 2}, 2\left(\gamma + \frac{1}{\ln 2}\right), \ldots, 200\left(\gamma + \frac{1}{\ln 2}\right)$. For each of those WORK points, we record a flag indicating whether the honest parties or the adversary first mined a chain with at least that amount of work. Similar to the Bitcoin experiments, we ran 100 Monte Carlo simulations and calculated k^* to be the WORK point such that at least 90% of the Monte Carlo simulations indicated that the honest parties were ahead of the adversary from that WORK point onwards. The confirmation delay (d) for each configuration g was obtained by dividing this k^* by $\frac{f}{\gamma + \frac{1}{\ln 2}}$ to calculate the time needed to get k^* work on average.

4.3 Experimental Results

Figure 2(a) illustrates the evolution of d across different g values. We identified the best operating conditions for a protocol as the point where the confirmation delay is minimized. In our simulations, Bitcoin achieved its lowest delay of 25.58Δ at a block rate of $g = 0.98$. On the other hand, at the same g, PoEM reached the minimal delay of 18.29Δ operating with $\gamma = 10$. This



(a) d vs g



(b) f vs g

Fig. 2: (a) Confirmation delay d (measured in network delay Δ units) vs. the block production rate g. PoEM typically demonstrates lower confirmation delay for any given value of g, and the difference becomes more pronounced at large g. (b) The relationship between the honest block production rate g and the honest chain growth rate f for Bitcoin and PoEM.

means for the configuration minimizing the delay of each protocol, PoEM had a $\frac{25.58-18.29}{25.58} = 28.5\%$ lower confirmation delay as compared to Bitcoin.

When either system operates at higher block production rate g, the honest chain grows faster (f also increases), but the confirmation delay d increases as well. We found that the relationships between g and f in Bitcoin and PoEM are comparable (Figure 2(b)). However, for a given delay, PoEM can safely operate at a higher block production rate g than Bitcoin, yielding an operation point with a higher honest chain growth rate f. In particular, we found that, for example at $d = 25.6\Delta$, Bitcoin gives a chain growth rate of $f = 0.43$, whereas PoEM, for a close delay of $d = 27.4\Delta$, gives a chain growth rate of $f = 0.5$, marking a 16.3% improvement in transaction throughput.

5 Analysis

We now prove the security of PoEM.

Definition 8 (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 $\textsc{work}(C_2[|C_1|:]) \geq s\tau$.*

Definition 9 (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 $[\alpha{:}\alpha + \ell] \lhd C$.*

Definition 10 (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 $[:-k] \lhd C_1 \preccurlyeq C_2$.*

In the analysis we are going to assume honest majority.

Definition 11 (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, then $A_{r,i,j} = \textsc{work}(H(B))$. 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 12 (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 intrinsic work generated by all adversarial party queries during round r: $Z_r = \sum_{i=n-t+1}^{n} \sum_{j=1}^{q} A_{r,i,j}$.

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 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 13 (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 14 (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.*

- $$(1-\epsilon)\,\mathbb{E}[\underline{X}(S)] < X(S) < (1+\epsilon)\,\mathbb{E}[\overline{X}(S)] \tag{1}$$
- $$(1-\epsilon)\,\mathbb{E}[Y(S)] < Y(S) \tag{2}$$
- $$Z(S) < (1+\epsilon)\,\mathbb{E}[Z(S)] \tag{3}$$
- *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 1, 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 $\textsc{work}(B) \in \mathbb{R}^+$ function (an ideal quantity unobservable by any Turing Machine, as it cannot process real-valued inputs), and its κ-bit *discrete approximation* $\widetilde{\textsc{work}}(B)$ (observable by a Turing Machine by invoking $H(B)[:\kappa]$). 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 real-valued random variables and, in particular, random variables distributed according to the *continuous* exponential distribution. These distributions lend themselves to easier tools than conducting a cumbersome analysis in the discrete domain. 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 approximate work of blocks and chains. Observe that for hash input A, we have $\widetilde{H}(A) \leq H(A)$ and for block B we have $\widetilde{\textsc{work}}(B) \geq \textsc{work}(B)$, and for blocks B_1, B_2 we have $\textsc{work}(B_1) \geq \textsc{work}(B_2) \rightarrow \widetilde{\textsc{work}}(B_1) \geq \widetilde{\textsc{work}}(B_2)$. Additionally, because $T = T[:\kappa]$, for any block B it holds that $\widetilde{H}(B) < T \leftrightarrow H(B) < T$.

Furthermore, taking approximate works preserves the order of blocks in the following fashion.

Lemma 2. *For all $A, B \in (0, 1)$, it holds that $\textsc{work}(A) \geq \textsc{work}(B) \rightarrow \widetilde{\textsc{work}}(A) \geq \widetilde{\textsc{work}}(B)$.*

Proof.

$$\begin{aligned}
&\textsc{work}(A) \geq \textsc{work}(B) \Rightarrow -\lg A \geq -\lg B \Rightarrow \lg A \leq \lg B \\
\Rightarrow &A \leq B \Rightarrow 2^\kappa A \leq 2^\kappa B \Rightarrow \lfloor 2^\kappa A \rfloor \leq \lfloor 2^\kappa B \rfloor \Rightarrow \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \leq \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa} \\
\Rightarrow &\lg \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \leq \lg \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa} \Rightarrow -\lg \frac{\lfloor 2^\kappa A \rfloor}{2^\kappa} \geq -\lg \frac{\lfloor 2^\kappa B \rfloor}{2^\kappa} \\
\Rightarrow &\widetilde{\textsc{work}}(A) \geq \widetilde{\textsc{work}}(B)
\end{aligned}$$

□

Showing that the order of *chains* is preserved under approximate work is a bit more involved, and we will work towards it next. Towards this, we observe that the approximate 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 $\widetilde{\text{WORK}}(B) - \text{WORK}(B) < 2^{-\kappa/2}$. The probability $\Pr[CLOSE]$ is overwhelming in κ.*

Proof. Consider the event E in which for all blocks B it holds that $H(B) > \frac{1}{2^{\kappa/2}}$. Let us calculate the probability of $\neg E$. For $\neg E$ to happen, at least one block must have $H(B) \leq \frac{1}{2^{\kappa/2}}$. For any block B, it holds that $\Pr[H(B) \leq \frac{1}{2^{\kappa/2}}] = \frac{1}{2^{\kappa/2}}$ (from the uniform distribution of $H(B)$ 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 $\Pr[\neg E] = \Pr[\exists B : H(B) \leq \frac{1}{2^{\kappa/2}}] \leq \sum_B \Pr[H(B) \leq \frac{1}{2^{\kappa/2}}] \leq \frac{nqL}{2^{\kappa/2}}$, which is negligible in κ, so E happens with overwhelming probability.

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

$$\begin{aligned}
&\widetilde{\text{WORK}}(B) - \text{WORK}(B) = -\lg \widetilde{H}(B) - (-\lg H(B)) \\
&< -\lg\left(H(B) - \frac{1}{2^\kappa}\right) - (-\lg H(B)) \\
&\leq -\lg\left(\frac{1}{2^{\kappa/2}} - \frac{1}{2^\kappa}\right) - \left(-\lg \frac{1}{2^{\kappa/2}}\right) \\
&= -\lg \frac{1 - 2^{-\kappa/2}}{2^{\kappa/2}} - \frac{\kappa}{2} \\
&= -\lg 1 - 2^{-\kappa/2} \leq -\ln 1 - 2^{-\kappa/2} \leq 2^{-\kappa/2}\,.
\end{aligned}$$

The first inequality stems from the fact that $\widetilde{H}(B)$ and $H(B)$ must differ by less than $\frac{1}{2^\kappa}$. The second inequality stems from the fact that the function $-\lg\left(x - \frac{1}{2^\kappa}\right) - (-\lg x)$ is decreasing for $x > \frac{1}{2^\kappa}$. □

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

Corollary 1 (Chain Work Approximation). *In a PoEM execution, the probability that all chains C have $\widetilde{\text{WORK}}(C) - \text{WORK}(C) < Lqn2^{-\kappa/2}$ is overwhelming in κ.*

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

$$\begin{aligned}
&\widetilde{\text{WORK}}(C) - \text{WORK}(C) = \sum_{B \in C} \widetilde{\text{WORK}}(B) - \text{WORK}(B) \\
&< \sum_{B \in C} 2^{-\kappa/2} = |C| 2^{-\kappa/2} \leq Lqn2^{-\kappa/2}\,.
\end{aligned}$$

□

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 $w = \textsc{work}(B), \widetilde{w} = \widetilde{\textsc{work}}(B)$, and let C be the chain it extends, let $w_1 = \textsc{work}(C), \widetilde{w}_1 = \widetilde{\textsc{work}}(C)$, and $w'_1 = \textsc{work}(CB), \widetilde{w}'_1 = \widetilde{\textsc{work}}(CB)$. Consider any other chain C_i that appears in the execution, and let $w_2 = \textsc{work}(C_i), \widetilde{w}_2 = \widetilde{\textsc{work}}(C_i)$. Let $BADRANGE_{j,i}$ denote the event that both $w_1 < w_2$, and, furthermore, either $w_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq w_1 + w < w_2$ or $w_2 < w_1 + w \leq w_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}$. 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 w is distributed as $\mathsf{Exp}(\ln 2)$, so for every other chain C_i in the execution for which $w_1 < w_2$ we have

$$\begin{aligned}
\Pr[w_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq w_1 + w < w_2 | w_1 < w_2] = \\
\Pr[w_2 - w_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq w < w_2 - w_1 | w_1 < w_2] = \\
(1 - 2^{-(w_2 - w_1)}) - (1 - 2^{-\left(w_2 - w_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}\right)}) = \\
2^{-\left(w_2 - w_1 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}\right)} - 2^{-(w_2 - w_1)} = \\
2^{-(w_2 - w_1)}(2^{\frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}} - 1) \leq 2^{\frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}} - 1 < \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\,.
\end{aligned}$$

The second relation is from the cumulative distribution function of the exponential distribution; the fifth relation is from the conditioning on $w_1 < w_2$, and the last relation is from Lemma 11, noting that $0 < \frac{nqL+1}{2^{\kappa/2}} < 1$.

Similarly, for the other direction,

$$\begin{aligned}
\Pr[w_2 < w_1 + w \leq w_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | w_1 < w_2] = \\
\Pr[w_2 - w_1 < w \leq w_2 - w_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | w_1 < w_2] = \\
(1 - 2^{-\left(w_2 - w_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\right)}) - (1 - 2^{-(w_2 - w_1)}) = \\
2^{-(w_2 - w_1)} - 2^{-\left(w_2 - w_1 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\right)} = \\
2^{-(w_2 - w_1)}(1 - 2^{-\frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}}) \leq 1 - 2^{-\frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}} < \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}}\,.
\end{aligned}$$

Consequently,

$$\begin{aligned}
&\Pr[\text{BADRANGE}_{j,i}] = \Pr[\text{BADRANGE}_{j,i}|w_1 < w_2]\Pr[w_1 < w_2] \\
\leq &\Pr[\text{BADRANGE}_{j,i}|w_1 < w_2] \\
= &\Pr[w_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \leq w_1 + w < w_2 | w_1 < w_2] + \\
&\Pr[w_2 < w_1 + w \leq w_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} | w_1 < w_2] \\
= &2\frac{nqL+1}{2^{\kappa/2}}
\end{aligned}$$

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). *Consider a causal execution of PoEM. Let HS be the event that for all two (adversarial or honest) chains C_1, C_2 appearing in the execution, if $\textsc{work}(C_1) < \textsc{work}(C_2)$, then $\widetilde{\textsc{work}}(C_1) < \widetilde{\textsc{work}}(C_2)$. The probability that $\neg$HS is negligible 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 1 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_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 $w = \textsc{work}(B), w_1 = \textsc{work}(C_1), w_1' = \textsc{work}(C_1')$ and $\widetilde{w}, \widetilde{w}_1, \widetilde{w}_1'$ be the respective approximate works. Consider any other chain C_2 with work $w_2 = \textsc{work}(C_2)$ and approximate work $\widetilde{w}_2$ that has already appeared in the execution, and consider the undesirable event FLIP_{C_1,C_2} that $w_1' < w_2 \wedge \widetilde{w}_1' \geq \widetilde{w}_2$ or $w_1' > w_2 \wedge \widetilde{w}_1' \leq \widetilde{w}_2$. If $w_1 \geq w_2$, then, because $w > 0$, therefore $w_1 + w > w_2$ and hence $w_1' > w_2$. Additionally, by HS_{j-1} we have $\widetilde{w}_1 > \widetilde{w}_2$, therefore $\widetilde{w}_1 + \widetilde{w} > \widetilde{w}_2$, and $\widetilde{w}_1' > \widetilde{w}_2$. From this, it follows that $w_1 \geq w_2$ yields $\neg\text{FLIP}_{C_1,C_2}$. Thus, it suffices to only consider the situation where $w_1 < w_2$.

Case 1: $w_1 + w < w_2$. From the conditioning on ¬BADRANGE, we have $w_1 + w < w_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}}$, therefore $\widetilde{w}_1 - \frac{nqL}{2^{\kappa/2}} + w < w_2 - \frac{nqL}{2^{\kappa/2}} - \frac{1}{2^{\kappa/2}} \Rightarrow \widetilde{w}_1 + w < w_2 - \frac{1}{2^{\kappa/2}} \Rightarrow \widetilde{w}_1 + \widetilde{w} - \frac{1}{2^{\kappa/2}} < w_2 - \frac{1}{2^{\kappa/2}} \Rightarrow \widetilde{w}_1 + \widetilde{w} < w_2 \Rightarrow \widetilde{w}_1' < w_2 \leq \widetilde{w}_2$. The first inequality is obtained from the conditioning on CLOSE,

noting that $\widetilde{w}_1 - \frac{nqL}{2^{\kappa/2}} < w_1$ follows from $\widetilde{w}_1 - w_1 < Lqn2^{-\kappa/2}$ (Corollary 1). The third inequality is also obtained from the conditioning on CLOSE, noting that $\widetilde{w} - \frac{1}{2^{\kappa/2}} < w$ follows from $\widetilde{w} - w < 2^{-\kappa/2}$ (Lemma 3). It follows that $\neg\text{FLIP}_{C_1,C_2}$.

Case 2: $w_1 + w > w_2$. From the conditioning on $\neg$BADRANGE, we have $w_1 + w > w_2 + \frac{nqL}{2^{\kappa/2}} + \frac{1}{2^{\kappa/2}} > w_2 + \frac{nqL}{2^{\kappa/2}}$, therefore $w_1 + w > \widetilde{w}_2 - \frac{nqL}{2^{\kappa/2}} + \frac{nqL}{2^{\kappa/2}} \Rightarrow w_1 + w > \widetilde{w}_2 \Rightarrow \widetilde{w}_1 + \widetilde{w} > \widetilde{w}_2 \Rightarrow \widetilde{w}'_1 > \widetilde{w}_2$. 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 2 (Approximate Fork Choice). *In Hash Separated executions of PoEM, for any two chains C_1, C_2 it holds that $\widetilde{\text{WORK}}(C_1) < \widetilde{\text{WORK}}(C_2) \to \text{WORK}(C_1) < \text{WORK}(C_2)$.*

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

Theorem 1 (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 12 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 15 (Block Work Interval). *A block B of chain C has* work interval $I(B) = \{\xi \geq 0 : [\xi] \lhd C = B\}$.

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 $B' = \lceil \xi \rceil \lhd C'$ 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 work greater or equal to ξ.

Claim: Every block computed after round r will be extending a chain with 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, $\widetilde{\text{WORK}}(C^*) \geq \widetilde{\text{WORK}}(C)$. By Hash Separation, $\text{WORK}(C^*) \geq \text{WORK}(C) \geq \xi$.

If B is computed after round r, it holds that $\xi \notin I$ (noting that $\text{WORK}(B) > 0$). If B' is computed after round r, it holds that $B' \neq \lceil \xi \rceil \lhd C'$. Both lead to a contradiction. □

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

Proof. By induction on r_2. For the inductive base ($r_2 = r_1$), observe that if at round r_1 an honest party has a chain C of work w, then that party broadcasted C at the end or round $r_1 - 1$. It follows that every honest party receives C at round r_1 and adopts a chain with greater or equal work.

For the inductive step, note that by the inductive hypothesis, every honest party has received a chain of work at least $w' = w + \sum_{r=r_1}^{r_2-2} X_r$ by 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 successfully queried the random oracle with a chain of work at least $w' + X_{r_2-1}$ and broadcasted it to the network. At round r_2, every honest party receives the chain and adopts a chain of work at least $w' + X_{r_2-1} = w + \sum_{r=r_1}^{r_2-1} X_r$. □

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 14 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 14 Eq. 2, the second from the bound in Lemma 1 Eq. 4, and the last one from part (1) of this proof. □

Theorem 2 (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 14 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, $\textsc{work}(C_2) \geq \textsc{work}(C_1) + X(S)$, as desired. □

Lemma 9 (Entropic Patience). *In a typical execution, any chained 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 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 3. *In a typical execution of PoEM, for any honest party P and any round r it holds that* $\textsc{work}([:-k] \lhd {}^{P}\boldsymbol{C}_r) < 2k$.

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

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

□

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 $[:-k] \lhd C_1 \preccurlyeq C_2$ and $[:-k] \lhd C_2 \preccurlyeq C_1$ 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 $[:-k] \lhd C_1 \not\preccurlyeq C_2$ or $[:-k] \lhd C_2 \not\preccurlyeq C_1$. 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 work of blocks computed by honest parties during convergence opportunities in S with adversarial 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 work. Also note that $\textsc{work}(C_1) \geq \max \Xi$ and $\textsc{work}(C_2) \geq \max \Xi$ 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 $[\xi] \lhd C_1 \neq \perp$ and $[\xi] \lhd C_2 \neq \perp$.

We now argue that for every $\xi \in \Xi$ either block $[\xi] \lhd C_1$ or block $[\xi] \lhd C_2$ is adversarial. If the block lies on the common prefix of C_1 and C_2, namely $[\xi] \lhd C_1 = [\xi] \lhd C_2$, 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 $[\xi] \lhd C_1$ and $[\xi] \lhd C_2$ cannot both be honest. This completes the proof of the claim $Z(S) \geq Y(S)$.

All the chained work $\max(\textsc{work}(C_1[i^*:]), \textsc{work}(C_2[i^*:])) \geq k$ 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 3 (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 $[:-k] \lhd C_1 \not\preceq C_2$.

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 $[:-k] \lhd C_1 \preceq C$ 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, $\textsc{work}(C) \geq \textsc{work}(C_1)$ due to the heaviest chain rule followed by P_1. Therefore, $[:-k] \lhd C_1 \preceq [:-k] \lhd C$. By the Common Prefix lemma, we have $[:-k] \lhd C \preceq C_2$ (at r_2, party P_1 has C and party P_2 adopts C_2). By transitivity of $\preceq$, we have $[:-k] \lhd C_1 \preceq C_2$, 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 $[:-k] \lhd C_1 \preceq C$ (at $r_1 + 1$, party P_1 adopts C and has C_1). Furthermore, $\textsc{work}(C) \geq \textsc{work}(C_1)$ due to the heaviest chain rule followed by P_1. Because $\textsc{work}(C) \geq \textsc{work}(C_1)$, therefore $[:-k] \lhd C_1 \preceq [:-k] \lhd C$. By the Common Prefix lemma, we have that $[:-k] \lhd C \preceq C_2$ (at $r_1 + 1$, party P_1 adopts C and party P_2 has C_2). By transitivity of $\preceq$, we have $[:-k] \lhd C_1 \preceq C_2$, which contradicts the violation of Common Prefix. □

Theorem 4 (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 $\textsc{work}(C_1) \geq \ell$ and quality lower than μ, namely the sum x of works of all honestly generated blocks in C_1 is less than $\mu \textsc{work}(C_1)$. Consider the minimum 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 $L = \textsc{work}(C_2)$ 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 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 weight at least $\textsc{work}(C_3[u':])$ from round $r_1 + 1$ onwards. Therefore, by Lemma 7, at round r_2, every honest party adopts a chain of weight at least $\textsc{work}(C_3[u':]) + X(S)$. But we know that P_2 adopts a chain of length $\textsc{work}(C_3[u':]) + L$, 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. □

Theorem 5 (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 3), it follows that if $r_1 \leq r_2$, then $[:-k] \lhd C_1 \preccurlyeq C_2$; and if $r_2 \leq r_1$, then $[:-k] \lhd C_2 \preccurlyeq C_1$. In both cases, 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 6 (PoEM is Live). *Typical executions of PoEM are live with parameter* $u = \max(\left\lceil \frac{\ell+2k}{(1-\epsilon)f} \ln 2 \right\rceil, s)$.

Proof. Consider any round r. Because $u \geq s$, invoking Entropic Growth(s, τ) (Theorem 2), we conclude that for all honest parties P and all rounds $r' \geq r + u$, it holds that $\textsc{work}({}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]) \geq u\tau \geq \ell + 2k$. From Corollary 3, it follows that $\textsc{work}([:-k] \lhd {}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]) \geq \ell$. Invoking Entropic Quality (Theorem 4), it holds that in the chain segment $[:-k] \lhd {}^{P}\boldsymbol{C}_{r'}[|\,{}^{P}\boldsymbol{C}_{r}|:]$, 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 $\mathsf{tx} \in {}^{P}\boldsymbol{L}_{r'}$. □

Corollary 4 (PoEM is Secure). *PoEM is secure with overwhelming probability.*

Proof. Executions are typical with overwhelming probability (Theorem 1). Typical executions are safe (Theorem 5), and live (Theorem 6), from which security follows. □

6 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 hierarchical chain constructions 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 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 $\mathsf{Bern}(\cdot)(\gamma + \mathsf{Exp}(\ln 2))$ 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 parametrization triplet (g, γ, k) that minimizes the confirmation delay. Can this optimal parametrization be found analytically?

Additionally, we know that, at the operating limit of $\gamma \to \infty$, the protocol is exactly Bitcoin, so we have proofs of security for both $\gamma = 0$ and $\gamma \to \infty$, but not for $0 < \gamma < \infty$. We leave the formal analysis of these questions for future work.

Work functions. We used the function $\textsc{work}(B) = \gamma - \lg H(B)$. 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 delay and throughput.

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 [10]. Follow up works in Bitcoin [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]. 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.

7 Conclusion

In this paper, we introduced PoEM, a new fork choice rule, in which each block counts for $\text{WORK}(B) = -\lg \frac{H(B)}{T}$ instead of the usual $\text{WORK}(B) = \frac{1}{T}$ (Section 3). We illustrated experimentally (Section 4.3) that PoEM achieves better transaction confirmation latency (28.5% improvement), or better transaction throughput (16.3% improvement), for the same level of adversarial resilience ($\beta = \frac{t}{n}$) and level of security (security parameter κ). We formally proved the security of PoEM (Corollary 4) 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 and has seen wide community adoption (Section 4.1). Our protocol only changes the proof-of-work inequality, and thus is composable with a multitude of 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. 2024.

A 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 a given value of g, the delay of the system behaves as illustrated in Figure 3 for varying values of γ. We believe that this graph behaves convexly for small values of γ, whereas, for $\gamma \to \infty$, the delay converges to the behavior of Bitcoin. 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 amount of work, converging at the limit to the longest chain rule.

B Mathematical Background

Lemma 11. *For all $0 < y < 1$, it holds that $2^y - 1 < y$ and $1 - 2^{-y} < y$.*

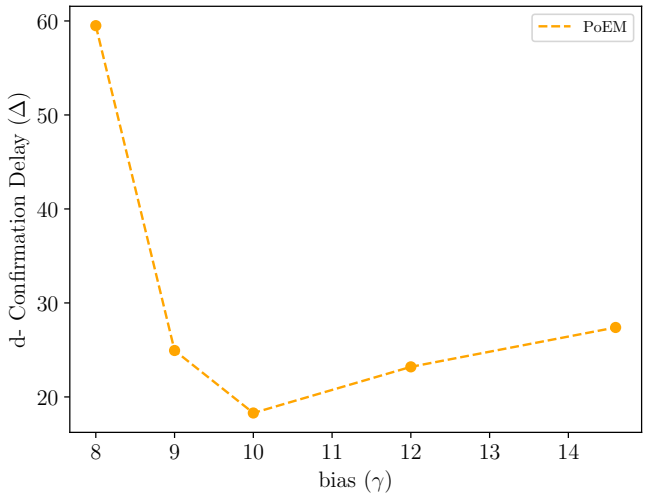


Fig. 3: Confirmation delay d (measured in network delay Δ units) vs. the bias γ. The bias, γ was experimentally optimized for the $g = 0.99$ which showed the lowest confirmation delay in Bitcoin.

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

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

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

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

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

For all $0 < t < \lambda$:

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

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

$$\frac{d}{dt}\ln\left[(1-p) + p\frac{\lambda}{\lambda - t}\right] - t(1+\epsilon)\frac{p}{\lambda} =$$
$$\frac{1}{(1-p) + p\frac{\lambda}{\lambda-t}}\frac{d}{dt}\left[(1-p) + p\frac{\lambda}{\lambda - t}\right] - (1+\epsilon)\frac{p}{\lambda} =$$
$$\frac{p\frac{\lambda}{(\lambda-t)^2}}{(1-p) + p\frac{\lambda}{\lambda-t}} - (1+\epsilon)\frac{p}{\lambda}$$

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

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

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 $t < 0$:

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

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

$$\frac{d}{dt}\ln\left[(1-p) + p\frac{\lambda}{\lambda - t}\right] - t(1-\epsilon)\frac{p}{\lambda} =$$
$$\frac{1}{(1-p) + p\frac{\lambda}{\lambda - t}}\frac{d}{dt}\left[(1-p) + p\frac{\lambda}{\lambda - t}\right] - (1-\epsilon)\frac{p}{\lambda} =$$
$$\frac{p\frac{\lambda}{(\lambda-t)^2}}{(1-p) + p\frac{\lambda}{\lambda - t}} - (1-\epsilon)\frac{p}{\lambda}$$

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

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

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 $\Pr[X < (1-\epsilon)\,\mathbb{E}[X]] < e^{-\Omega(n)}$. □