Proof-of-Time for Monero PoW: Achieving Instant Finality, Eliminating Re-orgs and Preventing Selfish Mining, Irrelevance of 51% Dominance (Adapt or Perish)

Abstract

The Proof-of-Time protocol augments the RandomX PoW model by binding a timechain subset model. In tandem with modest PoW model adjustments, it achieves instant finality, eliminates block reorganizations and selfish mining, and defends community tenets by averting the PoW model from becoming subservient to PoS, precluding network and community split.

Proof-of-Time introduces a 30-second grace period deadlines derived from RandomX hash outputs bound to the public key, allowing miners to compete effectively with lower-deadline solutions. The system supports both solo mining and shared mining models, enabling proportional rewards and optional hashrate sampling, without bloating the blockchain and eliminating the need for pools. Through decentralized time consensus, without the need for OS clock synchronization, it renders 51% dominance inconsequential and features a maximum block time that prevents chain death.

Drawing from a reference ANNE datachain Proof-of-Spacetime (PoST) implementation, the Proof-of-Time model presents a decentralized timekeeping system that incentivizes network participation, offering a secure, lightweight, low-bandwidth-compatible solution with no network-wide aftermath rollbacks.

Time emerges as an integral aspect of PoW through probabilistic block production, empowering the network to self-govern and make decisions before it acts.

1. Introduction

The "longest chain" principle in Proof-of-Work consensus, albeit foundational to many blockchains, represents a probabilistic mechanism that introduces inherent vulnerabilities, including block reorganizations, selfish mining, and susceptibility to 51% attacks, thereby necessitating dependence on external validation layers to achieve practical finality. It is far from an unassailable feature. Such an approach represents a structural limitation of asynchronous mining, where chain selection favors propagation speed over deterministic security, compromising network autonomy, stability, and reliability.

The Monero blockchain can become a self-regulating entity, overcoming these challenges. This Proof-of-Time (PoT) proposal delivers a strategic response, addresses this constriction directly within the PoW framework by integrating hash-derived deadlines and a 30-second grace period atop the RandomX algorithm, enabling instant finality through network-wide selection of the lowest virtual-time solution, thereby eliminating re-org risks. PoT supplies a structural opportunity in asynchronous mining, where chain selection prioritizes speed of propagation alongside deterministic security, augmenting network autonomy without superseding Monero's Proof-of-Work or its core egalitarian mining ethos.

This proposal serves as a blueprint for implementation, documenting technical and incentive specifications that collectively provide a comprehensive, actionable framework for developers. We explore how PoT integrates with existing architecture, assess its security against network threats, highlight its potential to bolster stability and reliability, and position it as an evolution of the PoW consensus model, reinforcing the longest chain rule with security intrinsic to the PoW protocol.

2. Proof-of-Time Protocol: Conceptual Framework

PoT integrates a timechain model into the PoW blockchain, wherein time is an inherent feature of the PoW mining process itself, governed by network time consensus, rather than universal clocks, blended with PoW consensus. Nodes auto-adjust their time offsets based on peer reports, and miners compete to submit solutions with the lowest deadlines within a defined timeframe.

2.1 Timechain Concept

Endogenous time is calculated based on the local system clock, adjusted to blockchain genesis time, and a dynamic offset derived from peer interactions.

No centralized time source or OS clock synchronization is required. Nodes achieve consensus through peer reports from their connected subset (e.g., 10-20 peers in a 10,000-node mesh), forming transient segments that converge asynchronously via P2P propagation. In such large-scale P2P networks, all views are inherently partial by design. Each node samples only its direct connections, with global alignment emerging via gossip over 3-5 hops. Each node computes its time using this formula:

chain_time = floor((system_time - MONERO_EPOCH + 500 + ms_adjust) / 1000)

Where:

1. system_time is the current time in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC),
2. MONERO_EPOCH is April 18, 2014, 19:00:00 UTC (genesis time),
3. 500 ms provides a rounding offset to align seconds (biasing toward the nearest whole second for precision in integer division)
4. The variable "ms_adjust" represents a millisecond offset that is updated based on the binned mode approximation of the median time reported by connected peers (recommended for partial P2P views), or the direct median for full-sample scenarios. This mechanism facilitates decentralized synchronization. Typically, the offset can range from -86,400,000 milliseconds (representing a day) to much larger values, such as ±31,536,000,000 milliseconds (representing a year). This range is managed via a 64-bit long to ensure robustness against significant clock drifts. Updates are additive, preserving prior adjustments via ms_adjust += delta. It prevents oscillation or over-

correction in noisy networks, allowing incremental corrections over multiple recalibrations without resets to zero.

Nodes sustain a chain time offset extrapolated from at least five peer reports (prioritizing well-connected, non-rebroadcast peers sorted by clock diff reliability). The offset recalibrates when a significant majority of peers (e.g., 70%) indicate deviation (e.g., improbably high >10 seconds; categorized by diff thresholds: "potentially problematic" if >1999 ms, "borderline" if ≥500 ms, "acceptable" if <500 ms), securing time alignment and preventing network fragmentation, with ms_adjust computed via the recommended binned mode (detailed below) or the direct median formula:

ms_adjust = (median_peer_chain_time * 1000) - (system_time - MONERO_EPOCH + 500)

Binned Mode is best for partial views. Collect ms-level clock diffs from problematic peers - absolute value of diff (|diff| ≥500 ms), round each to the nearest 500 ms bin, increment frequencies in the bin and its neighbors (±500 ms, ±1,000 ms for smoothing), then set ms_adjust += -mode_bin (mode = most frequent bin). This cluster offsets robustly in limited samples (e.g., 10-20 peers), approximating the network median without complete chain_time data and handling noise from gossip delays.

Thereby, the individual "chain_time" matches the network median even for grossly erroneous local clocks. The binned mode offers a robust approximation in the standard partial P2P scenario, such as using message differences from 10 to 20 peers within a 10,000-node network. Over iterations, it converges to the direct median, which provides precise alignment with complete second-level samples. Action thresholds require a minimum of five problematic instances out of the total peer group, along with a minimum threshold of 70% to initiate any adjustments. Assessing at least five samples against the threshold strikes a balance between early detection and evidence strength, and effectively filters out noise while enabling a robust synchronization process.

If 70% or more of the peers demonstrate a "good enough" alignment, no further action is required. Otherwise, the system will recommend self-adjustment through DataGrouping binned mode analysis or the direct median formula above, provided that the necessary conditions are satisfied.

The recalibration procedure accounts for propagation delays (an extreme example, 15 seconds - see 2.2, 2.3, 4.1, the intents, due to their negligible size ~150 bytes, propagate within a small delta (e.g., <5 seconds)), and the 30-second grace period, maintaining consensus within a practical range. The system leverages peer connectivity and produces a perfect alignment without the need for operating system intervention (e.g., in a simulation with five peers reporting chain_times [329467195, 329467198, 329467200, 329467203, 329467208] s, the direct median is 329467200 seconds; a node with +4s drift adjusts ms_adjust to -4500 ms, yielding chain_time = 329467200s - aligning precisely without OS intervention. Using binned mode on corresponding diffs [-5000, -2000, 0, +3000, +8000] ms, problematic bins (-5000, -2000, +3000, +8000) smooth to cluster around 0 ms, yielding approximate ms_adjust ≈0 ms and refining iteratively. If ≥70% peers show "good enough" alignment, no action; otherwise, recommend self-adjustment via binned mode or direct median analysis if conditions met.

Peer	Input Drift (ms)	Pre-Adjustment chain_time (s)	Median Consensus chain_time (s)	ms_adjust (ms)	Post-Adjustment chain_time (s)
P1	-86,400,000 (-1 day)	329,380,800	329,467,200	86,400,000	329,467,200
P2	-3,600,000 (-1 hour)	329,463,600	329,467,200	3,600,000	329,467,200
P3	-5,000	329,467,195	329,467,200	5,000	329,467,200
P4	-2,000	329,467,198	329,467,200	2,000	329,467,200
P5	0	329,467,200	329,467,200	0	329,467,200
P6	3,000	329,467,203	329,467,200	-3,000	329,467,200
P7	8,000	329,467,208	329,467,200	-8,000	329,467,200
P8	+3,600,000 (+1 hour)	329,470,800	329,467,200	-3,600,000	329,467,200
P9	+86,400,000 (+1 day)	329,553,600	329,467,200	-86,400,000	329,467,200
P10	+31,557,600,000 (+1 year)	361,024,800	329,467,200	-31,557,600,000	329,467,200

This design delivers time-consensus that is asynchronous, lightweight, resilient, and fully decentralized. Nodes "vote" via medians, not elections.

2.1.1 Timechain Alignment Assessment

During the initial 1-3 block intervals (roughly 2-6 minutes) before complete median stabilization through the whole Monero network, local segments might experience transient time variances of a theoretical extreme example ~10-15 seconds due to asynchronous P2P propagation and partial peer views, but this has a negligible impact thanks to the protocol's built-in buffers and deterministic design. Miners with grossly erroneous local clocks cannot submit valid intents until their chain time aligns. No blocks are invalidated, re-orgs are impossible, and mining/intent submission remains fair and continuous.

What Happens in the Transient Phase (1-3 Blocks):

Local Segment Formation and Variance:

Nodes start with their initial chain_time (based on local ms_adjust), forming "segments" of 10-20 peers. In a 10k-node mesh, these partial segments enable scalable gossip, with variances arising from hop delays but resolving rapidly. Early medians may differ slightly (e.g., Segment A at 329,467,200 seconds, Segment B at 329,467,205 seconds) because reports haven't fully gossiped across the ~3-5 hop diameter of the network.

Buffered Operations During Variance:

Miners submit their intent deadlines based on local "chain_time." The 30-second grace period, initiated upon the first valid intent, accommodates even improbably high worst-case scenario propagation delays of ~10-15 seconds and a segment variance of ~10-15 seconds. Late intents from desynced segments are still accepted if within the window relative to the last block's embedded timestamp (which carries the producer's median-aligned time). As a result, there are no missed blocks - all competitive solutions vie fairly.

The winning block's timestamp is set by the miner's adjusted chain time, serving as a reliable report from peers. Nodes use this data to refine their consensus through median recalibrations. Subsequent blocks propagate this aligned timestamp, accelerating network-wide convergence (e.g., after Block N, ~50% of the network incorporates the adjustment; by N+2, ~90%).

Nonce iteration and difficulty targets remain unaffected - variances don't alter hash validity or rewards, as deadlines normalize to the local view but rank network-wide by absolute value.

Impact on a Large Network

In a 10k-node net, <1% of intents might arrive 5-10s "late" in transient segments, but the grace period rejects only those >30s off the triggering intent, preventing exploits. No chain forks occur because PoT's instant finality deterministically selects the lowest deadline per block, while the variances only slightly shuffle the submission order without affecting the outcomes.

Self-healing convergence occurs rapidly. Gossip-based P2P models, as simulated in ns-3 for blockchain networks, demonstrate 80-90% alignment within one block (approximately 2 minutes) and wide alignment (with less than 2 seconds of global variance) by three blocks, as the medians average out via overlapping segments. The recommended binned mode further enhances this in partial samples by clustering problematic ms diffs into 500 ms bins with neighbor smoothing, approximating the direct median and accelerating convergence in noisy conditions (e.g., mode of binned diffs yields ~0 ms adjust in balanced transients, even with 10-20 peer views). In a worst-case scenario, a partitioned 20% of the net will self-adjust upon reconnection, buffered by a maximum block time of 2.5 minutes to prevent stalls.

1. From **"Improvements of Blockchain's Block Broadcasting: An Incentive-Based Approach"** (ePrint 2018/1152, Authors: Qingzhao Zhang et al. - https://eprint.iacr.org/2018/1152.pdf). This paper uses ns-3 simulations to model P2P block propagation in a 1000-node network, showing fast convergence via block sharding. A technique that divides data into small pieces for parallel relay across local segments, analogous to PoT's lightweight intent gossip (~150 bytes) propagating deadlines and hashes through overlapping peer subsets for rapid time consensus.

- Page 1 (Section I, Introduction): "In our simulation in Section VII-A, block sharding shortens the synchronization latency by about 90%." - demonstrates rapid latency reduction for network-wide alignment, implying 80-90% faster convergence in early blocks - mirroring the 80-90% alignment within one block in PoT's transient phase as intents gossip across 3-5 hops in partial views.
- Page 8 (Section VII-A, Simulation Results): "Compared with the baseline, block sharding can speed up the synchronization 30 times and shorten the time cost by over 90%." - quantifies convergence speedup, with 90%+ efficiency gains after initial propagation, supporting <2s global variance by block 3 in segmented networks via redundant paths that self-heal partitions, much like PoT's additive adjustments over gossip iterations.

2. From **"Computer Network Simulation with ns-3: A Systematic Literature Review"** (Electronics 2020, 9(2), 272; Authors: Campanile et al. - https://www.mdpi.com/2079-9292/9/2/272) This review surveys 128 ns-3 papers on P2P/blockchain simulations, highlighting convergence in gossip-based protocols (e.g., epidemic routing for alignment).

- Page 18 (Reference [6] at https://dl.acm.org/doi/pdf/10.1145/3199902.3199907): "Implementation of epidemic routing with IP convergence layer in ns-3 [6]..." The simulations in the cited work [6,128] demonstrate fast convergence, with protocols reaching ~90% message delivery within 3-5 hops and latency under 2 seconds in 1000-node P2P setups (scaled from smaller dynamic network tests) - "Fast convergence" in

epidemic (gossip) models aligns with PoT's P2P propagation, where 80-90% node alignment occurs within 1-2 intervals, as in ns-3 tests for overlapping segments.

- Page 25 (Reference [128] at https://resilinets.org/papers/Alenazi-Cheng-Zhang-Sterbenz-Epidemic-2015.pdf): "Epidemic routing protocol implementation in ns-3 [128]"... Simulations [in Alenazi et al.] achieve 100% message delivery across 1-8 hops, with latencies dropping ~80% in 10-50 node dynamic P2P setups, supporting the 80-90% alignment by first block and <2 s variance by block 3, via hop-based gossip in partitioned nets, directly analogous to PoT's median averaging over 3-5 hops.

Like Bitcoin's orphan rate (<0.1% from propagation delays), Monero's historical re-orgs are rare (~1-2 blocks deep, <1% frequency), and PoT eliminates them. The transient time wobble adds zero extra risk.

2.2 RandomX Hash-Derived Deadlines

We propose a nominal adaptation to the Monero mining system inspired by the PoST nonce-derived deadlines concept. Deadlines can be derived from PoW RandomX and computed in real-time, as seconds. A deadline represents an estimated time elapsed from the last block timestamp, where lower values indicate higher-quality solutions relative to the current difficulty.

The valid RandomX hash result converts to a deadline by normalizing its quality relative to the difficulty target and scaling to a maximum range:

deadline = floor((RandomX_hash_as_256bit_int × 180) / target)

The target is the current network difficulty target (the maximum acceptable 256-bit hash value).

To prevent turbo blocks and excessive delays, we clamp the deadline:

deadline = min(180, max(90, floor((RandomX_hash_as_256bit_int × 180) / target)))

The formula normalizes hash quality to produce a pseudo-time value, simulating a probabilistic "mining race" without real clocks, creating a natural ranking system for contending solutions. Valid hashes are uniformly spread within the range [0, target], resulting in deadlines that are too uniformly spread between [0, 180] seconds, regardless of the current difficulty.

If the difficulty doubles, it means the target halves, and the expected time to find a valid solution also doubles. However, the relative quality of the solutions found and the distribution of deadlines remain consistent. This consistency helps maintain fairness. Block times, therefore, stay on target at around 2 minutes, with the winning deadline typically at 90 seconds.

Lower deadlines reflect smaller RandomX hash outputs relative to the target, incentivizing rapid nonce iteration without altering core hash computations. The 90-second floor mandates that even exceptionally lucky hashes (near-perfect quality) map to no lower than the network target block time, preventing turbo blocks by enforcing a minimum deadline threshold while keeping such solutions competitively superior (as they tie for the lowest possible value).

The deadline is valid if the underlying hash meets the network difficulty. If a deadline falls below a configurable target (global or miner-specific), the miner submits their solution, deadline, and public key via RPC.

Public key-bound hashes and deadlines secure exclusive submission rights for the miner. Signing the submission with a private key or a seed stored securely in an encrypted local node config warrants its security. Any alteration of the solution triggers a mismatch with the public key or height, resulting in a protocol-level rejection that preserves solution integrity.

The network verifies the RandomX hash output against the difficulty target, recomputing the deadline to detect tampering.

Nodes reconfirm the RandomX hash and deadline, computing compliance against the difficulty target and miner identity. Peers reject duplicate submissions from the same public key, enforcing the one-solution rule, with valid intents held in a temporary pool until the grace period ends.

In cases of identical deadlines, the block with the lowest block hash breaks the tie, providing a deterministic and equitable resolution.

The difficulty adjustments control the expected lowest achievable deadline, targeting 2-minute block times, which enhances fairness by ranking solutions temporally without altering core hash computations.

2.3 Grace Period and Instant Finality

A 30-second grace period triggers with the submission of the first valid solution intent (hash, deadline, public key), which miners may broadcast up to 30 seconds before their deadline elapses from the last block timestamp.

This window accommodates network propagation delays, allowing all miners a fair chance to submit. Nodes retain these intents until the period concludes, selecting the solution with the lowest deadline, while rejecting late submissions to force submission and transparency.

Nodes automatically discard any worse (higher-deadline) solutions received, as they cannot compete with pending better ones. The timechain, with offsets adjusted based on the median time reported by a significant majority of peers (e.g., 70%), secures alignment with the honest chain, sustaining fairness and robustness, permitting for network latency and segmentation.

All peers evaluate intents at the end of the 30-second window. It does not matter when the grace window actually starts on all the nodes. It does not have to last 30 seconds for all and be simultaneous across the whole network. This flexibility arises because the grace period is triggered individually by each node based on the last block timestamp and the deadline of the first received solution intent.

The solution intents propagate within a small delta (e.g., <5 seconds). Nodes calculate the remaining grace time relative to their local chain time offset, adjusted by peer reports, allowing peers to consider late-arriving intents if received before the 30-second window expires network-wide.

For instance, if a node receives a 90-second deadline submitted 60 seconds after the last block timestamp, it validates and broadcasts the intent, triggering the 30-second grace period immediately for the network at the first valid submission step. Even if a node receives the first

solution delayed, the system checks the submission timestamp as soon as it arrives, calculating remaining grace relative to its offset clock.

To further bulwark equitable network play and sufficient time for intent propagation, submissions must arrive with at least 30 seconds remaining until the deadline elapses (i.e., submission_time ≤ last_block_timestamp + deadline - 30); intents arriving later are invalid and discarded, averting miners from delaying broadcasts to exploit timing edges (e.g., a 90-second deadline submitted at the 80-second mark, leaving only 10 seconds, would be rejected).

Miners stop mining upon receiving a grace trigger, broadcast of the first solution, as continued nonce iteration yields worse deadlines due to elapsed time. In theory, a miner could persist with intensive computation and submit a better solution by the end of the grace period; however, this is detectable via submission patterns and penalized by network consensus (e.g., flagging greedy attempts as anomalous, leading to peer disconnection through cool-off periods or black-listing).

A dishonest miner who delays submission too close to the end of the grace period forks to an invalid single-node chain, losing all peers. No one can override victory or cause disagreements at the network-wide level. The difficulty targets a best-block time of 2 minutes, without cumulative difficulty.

This process, governed by the decentralized timechain, achieves instant finality, rewarding the miner with the lowest deadline and eliminating selfish mining and re-org vulnerabilities.

2.4 Maximum Block Time

To prevent chain death and guarantee continuity, PoT sets a maximum block time of 2.5 minutes, aiming for a 2-minute target, even under extreme conditions. Mining starts immediately after the prior block is finalized and continues uninterrupted on the current block template until the grace period is triggered.

Miners broadcast valid solution intents (hashes meeting the adjusted difficulty threshold, where the effective target = (max_deadline_threshold / 180) × base_target) as they find them. The grace period begins only when the first valid intent is received, with a deadline that is less than or equal to the time elapsed since the last block timestamp.

The maximum acceptable deadline threshold scales linearly with elapsed time to accommodate low hashrate conditions:

max_deadline_threshold = min(270, t_elapsed + 45)

Where t_elapsed is the chain_time since the last block timestamp. This adjustment is computed deterministically by all nodes using the decentralized time consensus.

Examples:

1. At t_elapsed = 45s: threshold = 135s (effective target = (3/4) × base_target; early acceptance only for above-average solutions)
2. At t_elapsed = 90s (1.5-min target): threshold = 180s (effective target = base_target; normal operation)
3. At t_elapsed = 150s (2.5-min max): threshold = 195s (effective target = (11/10) × base_target; accepts inferior solutions)
4. Hard cap at 270s (4.5 minutes total, emergency liveness)

For instance, at 2.5 minutes (t_elapsed = 150 seconds), the threshold rises to 195 seconds, so a miner finding a hash that would convert to a 170-second deadline (rejected under the 180s cap) produce a 170-second deadline under the base target (i.e., base_target < hash ≤ effective_target) would now be accepted as valid, triggering the grace period and allowing competition among other arriving intents up to that threshold. Once a valid block is produced (via the lowest deadline intent within the threshold, with grace if triggered), the threshold resets to operate within the standard 90-180 seconds range for the next block.

This mechanism preserves existing difficulty adjustment for base hash requirements but adds a deadline acceptance window that expands during delays.

It averts stagnation without rollbacks or reorganizations by accepting progressively inferior yet still competitive solutions. Continuous operation of the honest chain is guaranteed even during extended periods of low activity. For example, hashrate drops of over 90%, while targeting winning deadlines of 90 seconds under normal 2-minute block time conditions.

3. Deadline-based Solo and Share Mining

We propose to reinforce the egalitarian mining ethos with a mixed mining model. The goal is to abandon the need for and use of centralized mining pools by dividing miners into two groups: solo miners and shared miners.

The coinbase will be revised to support multi-output.

3.0.1 Prior considerations - P2P network protections

To strengthen defenses against sybil attacks and spam, including potential key rotation by high-hashrate miners, the P2P layer enforces a strict limit of one intent per peer submitted strictly via localhost connection, applies ban scores to nodes submitting excessive intents, and curates peer connections to prevent flooding, making rotation and multi-intent blasting resource-intensive and unprofitable without meaningful gains in odds.

To diminish Sybil risks from attackers running multiple nodes attempting key rotation, a node reputation system based on connection duration and historical behavior solves the concern. We can use the rank to multiply the deadlines for new or low-reputation nodes.

3.1 Solo mining

Solo mining remains a core option, where individual miners compete to find valid RandomX hashes and derive low deadlines, submitting intents (hash, public key, deadline) via mining before the grace period triggers. Nodes collect and verify these intents within the 30-second grace period to select the lowest-deadline solution and claim block rewards.

Every 5th block, the solo miner with the lowest deadline wins the full block reward.

For all other blocks, the solo miner with the lowest deadline wins half the reward.

3.1.1 Solo Mining Handicap

We propose a limit of 5 block wins or fewer for any solo miner within a rolling 10-block era. Once a miner achieves the set wins within the rolling era, they are given a handicap in the form of a deadline multiplier penalty, which applies for the duration of the block era. For example, all their submitted deadlines are automatically increased by 50%. This adjustment encourages players to engage in solo mining.

While it does not prevent a miner from winning more blocks within the rolling window, it significantly increases the difficulty due to the handicap. The handicap can be adjusted further based on any subsequent wins.

3.1.2 Solo Mining Hashrate Sampling

Applying deadline-derived hashrate sampling to solo mining, as detailed in section 3.2.3, can improve fairness by creating more accurate handicaps. By averaging these hashrate estimates over a period, we can adjust penalties for dominant players, promoting long-term fairness and minimizing luck-based disparities without complicating the lowest-deadline win mechanic.

Additionally, solo miner hashrate samples can enhance peer reliability scoring in the recovery mechanisms outlined in section 4.2.

3.2 Share Mining

We offer three deadline-based share mining models for consideration. Each has a varying degree of simplicity, fairness, and security. All three options eliminate the need for second-layer pools at a low cost.

All three models prioritize low-hashrate participants, deter high-hashrate dominance through single-intent submission, and an inverted handicap model empowers smaller operations and mitigates spam risks.

In all three models, the remaining half of the non-solo mining rounds split rewards between 21 shared miners.

Share mining deadlines are not capped or clamped.

We recommend implementing the model discussed in section 3.2.3 below. Although it may not be the simplest option, it is provably fair and allows for a granular level of customization.

3.2.1 Native Top-21 Solutions Model

Shared miners submit exactly one intent per block round, choosing their best solution before the next block finalizes. The network ranks all submitted intents by deadline, where lower values indicate stronger solutions.

The open deadline range accepts solutions from 0 seconds upward without caps, allowing even slow solutions from modest hardware to compete as long as they rank in the top 21.

Nodes maintain a simple map of all single intents received, comparing new submissions against the current top-21 threshold. If a new deadline beats the 21st-place solution, it displaces the worst existing entry.

To level the playing field for smaller miners, the handicap targets those who repeatedly win, identified by their public keys. Nodes keep track of win counts over a rolling 10-block era.

If a miner achieves three or more wins within this window, cumulative penalties accrue for their subsequent rounds. The first time a miner reaches this threshold, their unclamped deadline multiplies by 3. In the case of a second offense, the penalty multiplier increases to 5, and so on.

If the handicapped deadline is greater than the current top-21 after intent submission, the solution is void. Penalties reset completely after 10 rolling rounds without any additional shared wins.

The model is the most lightweight option, relies entirely on the PoT deadline system for fairness, and promotes quality over quantity to determine rewards.

3.2.2 Unweighted Deadline Lottery

In this model, the 21 miners win through an unweighted lottery among all valid entries.

Shared miners submit exactly one intent per block round, with the network verifying that the hash meets the base difficulty and the deadline derivation is correct.

The lottery treats all valid entries equally, providing each participant with the same chance of winning, regardless of the quality of their submission. It utilizes a Verifiable Random Function (VRF) seeded by the hash of the previous block to select random 21 winners from the pool.

The open deadline range accepts solutions from 0 seconds upward without caps or lower thresholds, allowing even higher-derived deadlines from modest hardware to compete as long as they qualify as valid intents.

To level the playing field for smaller miners, the handicap targets those who repeatedly win, identified by their public keys. Nodes keep track of win counts over a rolling 10-block era.

If a miner achieves three wins within this window, all their subsequent solutions become void until the block era resets.

The effect is to reduce the competitiveness of high-hashrate miners in repeated wins, encouraging them to opt for solo mining where their advantages are less diluted. Low-hashrate miners gain relative ground without the quality thresholds that could exclude them.

The model relies entirely on the PoT deadline system for fair, verifiable competition. It encourages higher peer participation based on quantity rather than a quality model that determines rewards, empowering low-end hardware without centralized pools.

3.2.3 Weighted Deadline Lottery

This model draws inspiration from PRS workshares, replacing them with verifiable PoT deadlines for direct hashrate sampling, where rewards reflect genuine computational effort without the workshares overhead.

Shared miners submit exactly one intent per block round. The network verifies that the hash meets the base difficulty and the deadline calculation is correct, then derives an estimated hashrate sample using the formula:

estimated_hashrate = difficulty_target / (deadline × elapsed_time)

This raw estimated_hashrate serves as a proxy for the miner's effective computational contribution in that round: a lower deadline (indicating higher hash quality and thus "earlier" virtual solve time) implies a stronger sample, scaled by the actual elapsed time since the last block to account for varying mining durations. Individual values are noisy due to hashing luck, but they converge reliably over multiple samples.

The result provides a proxy for the relative contribution. A lower deadline indicates better hash quality, suggesting a higher effective hashrate. It is true even though individual samples can be noisy due to luck, and low-hashrate miners sometimes produce strong derived deadlines. Over a rolling 20-block era, nodes average these raw estimated_hashrate values to compute a close estimate per public key (converging to real hashrate within acceptable variance, estimated at 5-20%, depending on submission frequency and era luck). The greater the block era, the higher the degree of accuracy.

To prevent key rotation and reduce the exploitation of noisy single samples, a public key that has fewer than five samples in the rolling era will have its average estimated hashrate adjusted downward by multiplying the hashrate by (number of samples / 5). Such a method guarantees that new or rarely used keys receive proportionally fewer tickets, thereby reducing the incentive for sybil-based dominance.

For the lottery, the estimated relative hashrate (normalized against the network average) scales the ticket weight:

ticket_weight = max(1, estimated_relative_hashrate × 100)

The x100 is just a scaling knob to make the lottery more "proportional". The estimated relative hashrate is the average hashrate of a miner divided by the average hashrate of the network. It is determined by summing all participants' averaged hashrates and then dividing by the number of participants, rewarding higher average performance (e.g., a sample indicating 2× network share yields about 200 tickets) while providing a baseline for modest contributors (e.g., 0.1× yields 10 tickets).

The total ticket pool across all shared miners sets proportional odds, and a VRF, seeded by the previous block hash, randomly selects 21 winners from the pool, guaranteeing fairness and resistance to manipulation. We can prevent share mining hashrate inflation by limiting the total number of tickets per public key to 200.

An inverted handicap addresses serial winners more effectively. If a miner achieves three wins within a rolling 20-block period, all subsequent deadlines are limited to one ticket until the first win lapses over 20 blocks.

This model encourages high-hashrate miners to shift toward solo mining, where their strengths can shine without dilution, while keeping shared rounds accessible to smaller participants, and it relies entirely on the PoT deadline system for fairness and allows granular customizations for ticket scaling, era span, or high-end hashrate barring, while promoting decentralization through effort-based proportionality.

3.3 Assessment of Incentive

The three share mining models offer varied incentive structures to balance fairness, participation, and proportionality, all while integrating with the solo mining rewards (0.6 XMR block reward every 5th block, half otherwise) to promote peer decentralization.

The Native Top-21 Solutions Model incentivizes quality over quantity by directly rewarding the top 21 deadlines, favoring consistent high-hashrate miners with strong samples but potentially marginalizing low-hashrate participants unless they get lucky. The handicap encourages turnover, boosting small-miner opportunities through resets after 10 blocks.

The Unweighted Deadline Lottery promotes inclusivity by treating all valid submissions equally, offering the same chances regardless of the quality of their deadlines. This approach democratizes rewards for users with lower-end hardware, attracting more participants. However, it may diminish the importance of effort-based proportionality and could discourage intensive computation, which, however, is not essential within the scope of share mining.

Compared to the existing Monero model dominated by centralized pools (capturing ~90% of hashrate), these models are fairer for smaller miners overall, as solo cycles and deadline-based lotteries/top-selections offer significant payouts without 1–2% fees or centralization risks, with the weighted variant providing the most stable proportionality.

They all excel in decentralization by enforcing single intents, handicaps, and L1 participation to distribute power, outperforming pool concentration.

Although models introduce lottery variance in shared rounds, potentially reducing income stability compared to pool shares, we can mitigate it by solo guarantees and era averaging.

While the unweighted model risks under-rewarding effort, the top-21 model favors elites slightly more. The recommended Weighted Deadline Lottery strikes a proportional middle ground, using era-averaged hashrate estimates to scale tickets, rewarding sustained effort while capping at 200 to prevent dominance. It best aligns incentives with computational contribution, motivating upgrades without excluding modest setups.

All discussed mining models promote a more equitable ecosystem than the current pool-dominated system. They reward effort proportionally across hashrate levels while reducing systemic risks, such as 51% pool dominance.

3.4 Assessment of Security

Across the three share mining models, the PoT's core features collectively eliminate re-orgs, selfish mining, and withholding, while the maximum block time upholds liveness.

The Native Top-21 Solutions Model provides strong verifiability through direct deadline ranking and hash recomputation, with cumulative handicaps preventing serial dominance. Its quality emphasis could increase Sybil attack risk if high-hashrate actors rotate keys for multiple top entries; nonetheless, P2P curbs make this approach costly, requiring 100 to 400 intents and 100 to 400 Sybil nodes, which makes it unprofitable.

The Unweighted Deadline Lottery fights spam resistance by treating all participants equally and preventing more than three rewards within a rolling era. While its focus on quantity may lead to flooding, this risk is again mitigated by single-intent rules and VRF based on the previous block hash, making predictions difficult.

The Weighted Deadline Lottery strikes the most layered defense, with era-averaged hashrate sampling converging to 5–20% accuracy to normalize luck, ticket caps, and one-ticket handicaps post-three wins; the VRF lottery adds cryptographic fairness, while its noise tolerance prevents manipulation via randomly chosen samples.

In all models, high-hashrate key rotation remains theoretically possible but impractical due to P2P enforcement, accruing negative costs without odds gains, and handicaps dilute edges on winning keys.

The decentralized verification, which involves deadline recomputation, helps prevent forgery while grace periods are in place to ensure fair propagation.

Compared to the current Monero mining model, these changes make 51% dominance significantly more expensive, as deterministic finality secures reward locking on-chain without reliance on probabilistic confirmations or external dependencies. It creates a secure and fair framework that is resilient to centralization and exploitation.

4. Implementation Considerations

4.1 Time Collusion and Impact on Grace Period

If a majority of peers collude to report a fake time (e.g., an hour ahead), this could shift the ms_adjust value. It is not a relevant threat, as colluders would only harm themselves within their immediate peer bubble. Such skews misalign submissions, causing late or invalid intents that peers detect as greedy or hacked attempts (e.g., submitting quality solutions late in the grace period). In such an event, the bubble disconnects colluders, isolating them without affecting the rest of the network.

The core design's 30-second grace period, combined with the strict submission rules in sections 2.2 and 2.3 (e.g., rejecting late arrivals, grace triggers individually per node based on received intents and decentralized time offsets via median peer reports), provides ample buffer for typical network asynchrony.

Intents propagation, due to their negligible size of around 150 bytes, should occur globally in under 5 to 10 seconds, even on 90s-era and 56k modem bandwidth.

Withholding blocks is inherently futile because miners must broadcast intents early to trigger and participate in the grace period, with delays leading to rejection rather than overrides, shifting the paradigm from probabilistic longest-chain re-orgs to deterministic intent-based selection without history rewrites.

Clock adjustments weigh in a fuzzy, network-distributed manner:

Each peer's ms_adjust is the median made of offsets of connected peers, with outliers (e.g., reports deviating significantly from the majority) ignored to prevent skewing. If colluders form a tight-knit peer circle, they only alter their own isolated bubble, effectively forking themselves into irrelevance. If they infiltrate broader peer groups, they become the minority outliers and are discounted, as adjustments require consensus from a significant majority (e.g., 70%) of peers.

Miners gain nothing from skewing clocks, as grace triggers on first valid solutions and finality locks rewards regardless. Monero's highly distributed nature ensures honest segments dominate, rendering collusion impractical and self-defeating. No gain in rewards or attacks, only fragmentation and self-rejection.

The protocol can implement a hard cutoff to mitigate extreme cases. Peers can disregard an offset from a peer if it deviates by more than a predetermined threshold.

Empirical tests show that honest clock drifts rarely exceed a few seconds. Even if a miner's node starts with a time offset of 35 seconds, it will adjust to align with the honest majority through peer reports. This process ensures that submissions align correctly without disrupting the network.

4.2 Forked Peer Reconciliation and Recovery Mechanism

In rare instances of network partitions or temporary node isolation, we incorporate a controlled recovery function to enable affected nodes to detect divergence, validate the honest chain, and seamlessly rejoin it.

This mechanism upholds the PoT guarantee of no block re-orgs in the canonical chain under normal connected conditions. By treating finalized blocks instantly as immutable once integrated into the one chain, it limits recovery to shallow minority forks, thwarting abuse while permitting swift, non-disruptive reconnection without any risk of malicious history rewrites.

A core security pillar is robust eclipse attack resistance, achieved through a dynamic, multi-level quorum system. Nodes seek agreement not just from immediate level of connected peers, but via recursive queries across 2-3 hops to sample an effective ~200 peers network-wide (scaled proportionally for smaller chains, e.g., 20 for testnets), requiring a minimum 70% consensus on the alternative tip verified at a depth of three or more blocks to confirm chain integrity before any recovery action.

Divergence detection activates as nodes establish or re-establish connections with peers, leveraging Monero's established connectivity model. Nodes periodically query peers for the current chain tip (block hash and height) via a lightweight extension to existing P2P messaging.

If a peer reports a differing tip at the same or greater height, the node requests a concise chain summary, comprising block headers with timestamps, deadlines, and hashes, from the suspected forked node point onward. The node then traces backward until identifying a matching block hash, establishing the last common ancestor.

If no ancestor emerges within a practical depth of 100 blocks, the peer's chain is treated as incompatible, resulting in the disconnection of the peer and an increment to its ban score to deter spam or adversarial chain feeds. Deeper reconciliation in such cases requires manual intervention via "--allow-deep-recovery" config option.

We identify recovery targets for viable divergences by aggregating reports from a selected group of reliable peers. By calculating agreement levels based on their reliability scores, we obtain weighted agreement scores. These ensure a broad consensus and avert low-trust peers from disproportionately influencing decisions. Thus, the highest-scoring block serves as the recovery anchor for majority alignment with minimal data exchange.

Should multiple anchors tie in score, an extremely theoretical scenario arising solely from perfectly synchronized block production in isolated partitions, a tiebreaker selects the one with the greatest implied forward progress, based on the maximum reported height among agreeing peers. The peer-reported chain replaces the local peer chain only if its agreement score exceeds the local peer by a configurable margin of at least 10%, ensuring decisions resist minor hashing luck variances.

Recovery activation remains strictly conditional to maintain security - the peer-reported chain must fully validate under PoT rules, including hash compliance with difficulty, accurate deadline derivation, adherence to grace periods, and absence of duplicate intents.

During this process, the node re-verifies the entire post-fork segment sequentially, as if mining it in real-time, to confirm the chain's integrity against potential manipulation from the partition. Hash compliance ensures that each block's RandomX proof-of-work meets the era's difficulty target, adjusted for both forged and weakened hashes. Accurate deadline calculations confirm that the temporal ranking remains unchanged, preserving the integrity of the block with the lowest deadline. Adherence to grace periods confirms that intents are broadcast at least 30 seconds before the deadline, relative to timestamps and peer offsets. This measure prevents the use of withheld solutions that could lead to selfish mining and forks. The review for duplicate intents monitors key reuse, effectively invalidating any Sybil attack attempts that threaten

competition. If any compliance issues arise, the recovery process will be re-initiated, seeking agreement from further peers. Only those with a demonstrably honest history will be considered acceptable.

Auto recovery depth is restricted to forks no deeper than 10 blocks (measured from the common ancestor identified in the initial 100-block search), consistent with Monero's existing practices for shallow re-org handling. Deeper potential recoveries halt automation, logging a critical alert and requiring manual override via "--allow-deep-recovery" with explicit confirmation to mitigate contrived malicious chains. A temporal safeguard protects PoT's instant finality by prohibiting auto recovery for fork points older than one hour, ensuring that blocks finalized in real-time through the grace period's deterministic selection remain immutable in the canonical chain, even if a minority fork later attempts reconnection.

Recovery execution utilizes Monero's database primitives to revert the local state to the common ancestor, removing the peer's blocks, transactions, and associated state changes. The node then sequentially validates and appends the majority chain's blocks, recalibrating time offsets, deadlines, and rewards as needed, while reintegrating any compatible local pending transactions or intents. Upon completion, it broadcasts the updated tip to peers and resumes PoT mining on the refreshed block template, discarding any unsubmitted intents from the prior divergent path.

A command-line interface will support manual recovery oversight for operators.

This function extends Monero's existing sync architecture with PoT-aligned scoring, preserving backward compatibility for length-based decisions during transitional phases. Rigorous testnet evaluations, simulating partitions via delayed connections, will confirm recoveries complete in under one minute for shallow forks, reinforcing network convergence post-disruption while fully safeguarding against re-orgs and 51% threats.

4.3 Integration with Difficulty Adjustment Mechanisms

The PoT protocol closely interacts with Monero's difficulty adjustment algorithm, which aims for a 2-minute block time based on the direct median of recent block times.

To meet the target in the deadline-based system, we must algorithmically calibrate the base difficulty so that the expected lowest deadline across the network is about 90 seconds, based on a uniform distribution of deadlines between 0 and 180 seconds before clamping.

During implementation, the difficulty adjustment loop should use feedback from observed winning deadlines instead of relying solely on block timestamps. An increased difficulty can raise the distribution if the average deadlines trend below 90 seconds, indicating over-achievement.

Conversely, if deadlines frequently exceed 150 seconds or trigger the maximum block time threshold, difficulty decreases to encourage more valid solutions.

This approach ensures long-term stability and prevents drift from the target block time. Simulations using historical Monero hashrate data can fine-tune the adjustment parameters, avoiding oscillations. Compatibility is high, as the core difficulty formula remains unchanged, with PoT adding a deadline-derived modifier only during extended delays (via the max_deadline_threshold scaling).

4.4 P2P Network Modifications and Anti-Spam Measures

Implementing PoT requires enhancements to Monero's P2P layer to handle solution intent broadcasts efficiently and securely. While blocks already propagate via the existing "new block" messages, PoT introduces lightweight intents as precursors that broadcast before and during the grace period to collect competing solutions before the winning one assembles the block.

These intents are formatted as a new, compact message type (estimated ~150 bytes each, including the RandomX hash, deadline, public key, and signature), relayed similarly to transactions for rapid dissemination with minimal bandwidth overhead.

To prevent spam or sybil attacks, such as key rotation for multiple submissions, nodes must enforce strict rules: one intent per public key per block height, validated at receipt. Excessive submissions from a single peer trigger ban scores, with temporary disconnections for violators. Incoming intents are limited to one per connection every ten seconds during grace periods, and nodes only prioritize broadcasting those with deadlines better than the current known best.

These changes build on Monero's existing peer management, adding minimal complexity. Backward compatibility during a transition period could allow legacy nodes to observe but not participate in PoT mining, ensuring a smooth hard fork. Testing on the Monero testnet should verify propagation times remain under 5 seconds globally, aligning with the 30-second grace period.

4.5 Mining Software Compatibility

PoT is fully backward-compatible with existing RandomX mining software, requiring only minor updates to compute and submit deadlines from valid hashes. Miners would integrate a simple post-hash processing step: convert the hash to a 256-bit integer, apply the deadline formula, and clamp as specified. Submission occurs via RPC to the local node if the deadline meets a user-configurable threshold (e.g., below 150 seconds).

Miners benefit from the single-intent rule, which reduces the need for high-volume submissions and supports CPU-friendly configurations. Open-source miners, such as XMRig, can be enhanced with a PoT module. Workers refresh their block template every 30 seconds if no valid deadlines occur, and update the embedded timestamp to align with the latest chain_time (2.1) to prevent stale computations during ongoing nonce iteration. It updates the embedded timestamp to synchronize with the latest block time on the blockchain, which helps prevent stale computations during ongoing nonce iterations.

Implementation should include safeguards against tampering, such as signing intents with the miner's private key for verification.

4.6 Privacy and Security Implications for Miner Identities

PoT attaches deadlines to public keys, and in theory, the repeated use of the same key could allow observers to link mining activity across different blocks. The use of stealth addresses mitigates the risk through unlinkability of rewards in multi-output coinbases.

To further enhance privacy in this context, miners can leverage subaddress-derived keys for intent submissions. These ephemeral keys, generated from a secure seed, prevent long-term linkage without exposing the primary wallet, aligning with Monero's hierarchical deterministic structure and rendering activity tracking infeasible even under extended observation.

Security-wise, the system resists forgery through hash recomputation and signature checks, with network rejection of altered intents. We recommend key management in the node config. For example, encryption of the configuration file, with the node prompting users for a password on startup to decrypt and access private seeds securely, and encrypt the config again once loaded into memory.

4.7 Fork Deployment and Transition Strategy

As a consensus-altering upgrade, PoT necessitates a hard fork, scheduled at a predetermined block height to allow ample preparation.

During transition, a hybrid mode could permit legacy PoW blocks alongside the PoT model, with nodes preferring PoT-finalized blocks after activation.

After the fork, monitoring tools should track triggers of grace periods, distributions of deadlines, and applications for handicaps to detect any anomalies and prompt necessary adjustments to ensure that the correct scaling conditions were applied. This structured approach minimizes disruption, drawing on the history of successful upgrades such as RandomX integration.

4.8 Performance and Scalability Impacts

PoT introduces a lightweight overhead, which includes storing intents in a temporary pool (potentially reaching a few thousand per block during periods of high participation) and clearing after a grace period. The computational costs for deadline verification are minimal, especially when compared to the costs associated with RandomX hashing. Bandwidth usage remains low, with intents contributing approximately 1-2% to typical P2P traffic.

Scalability benefits include elimination of re-org processing, as instant finality averts chain rollbacks, improving node sync times. Benchmarks on varied hardware (e.g., Raspberry Pi for low-end nodes) should confirm no regressions in sync speed or resource usage, positioning PoT as a net positive for long-term network growth.

5. Conclusion

PoT marries with PoW through native integration, and together foster a far more equitable ecosystem, eradicating centralization risks from pools and rendering traditional 51% attacks irrelevant by eliminating re-orgs and selfish mining. Attackers with the majority of the hashrate can only mine forward at a probable loss under the handicap system, without the ability to rewrite history or censor, thus enhancing network resilience through deterministic finality and transparent intents.

Its simplicity supports low-bandwidth resilience, while Monero connectivity secures time consensus. Security analysis demonstrates that tamper-resistant deadlines and rejected colluding segments render timing collusions ineffective, and pose only a negligible risk and no ability to affect the overall network. Replacing cumulative difficulty with a maximum block time ensures liveness without rollbacks.

Overall, Monero has the opportunity to become the leader of the resurgence of PoW blockchains and prove once again that PoW is far superior to PoS through resilient, deterministic enhancements.

Acknowledgments

We thank the Monero community in advance for their input and the developers for considering this proposal.

Primary credit goes to Mr. Scatman of the ANNE Network for devising and implementing the PoST protocol, which facilitated this transition to PoT.

The PRS framework, developed by Dr. Karl Kreder of the QUAI Network, inspired the Weighted Deadline Lottery model and warrants recognition.

Limitations

This study is provided free of charge, as is, without guarantees of any kind, expressed or implied. Although the PoST is fully implemented, tested, and deployed on the ANNE mainnet, the authors of this proposal might lack comprehensive knowledge of the Monero protocol, and there are no research papers available on PoST. In light of the Monero situation, we endeavored to transpose the PoST model and document the PoT model to the best of our ability.

There might be solvable imprecisions or nuances beyond the scope of this proposal. Such challenges, through further research and revisions, can be resolved. The proposal achieves its fundamental purpose by evidently demonstrating that deterministic security is perfectly solvable within the native scope of PoW. It is delivered to encourage an in-depth exploration of technical feasibility and adaptability to Monero.

Research Bounty

A substantial, compelling, and negotiable bounty awaits one or more Monero developers who will conduct in-depth research into the technical feasibility of implementing the Proof-of-Time model into Monero under the following conditions.

1. Thorough research will demonstrate the technical feasibility and deliver necessary adaptations to this proposal.
2. Monero developers will reach a consensus and agree on the PoT implementation.
3. The Monero community will reach consensus.
4. A CCS implementation proposal reaches the Funding Required step.
5. A complete, partial, or adapted implementation of this proposal will qualify for the full bounty, as long as any form of PoS research and implementation is proven redundant and unequivocally rejected.

Once all conditions aggregate, the bounty will be due, paid as a lump sum that has been negotiated and agreed upon in advance. For more info, please message me on Matrix.

Meanwhile, we offer free and ongoing consultation to any Monero developer who aims to solve any challenges that may arise during the research and implementation process.