Visibility-Aware GHOST: Mitigating Visibility Asymmetry in Subtree-Based Proof-of-Work Consensus

Abdulwahab Almusailem[1 [0009−0001−5683−9599]⋆], Othman Alenezi[1 [0009−0001−0901−0761]], and Ameer Mohammed[1 [0000−0002−9494−8809]]

Department of Computer Engineering, Kuwait University, Kuwait
abd.almsailem@ku.edu.kw, a.othman@ku.edu.kw, ameer.mohammed@ku.edu.kw

Abstract. Subtree-based fork-choice rules such as GHOST can improve the throughput of Proof-of-Work (PoW) blockchains by crediting stale blocks, but they assume that honest miners have sufficiently similar views of the block tree. In heterogeneous networks, visibility asymmetry causes honest miners to compute different subtree weights, which can increase head oscillation, distort rewards, and create leverage for strategically timed releases. We propose Visibility-Aware GHOST (VA-GHOST), a drop-in fork-choice rule that estimates each block's visibility from lightweight decentralized attestations carried in subsequent blocks, assigns a visibility factor $\nu_b \in [0, 1]$, and computes subtree weight as a visibility-weighted subtree score. The empirical study is conducted on a node-local DAG implementation that realizes the protocol semantics of VA-GHOST, including local visibility counters, bounded certificates, and greedy visibility-weighted subtree selection. Across four scenarios, VA-GHOST substantially reduces reorganization frequency relative to LCR and standard GHOST. At the same time, it yields deeper average reorganizations, only modest fairness differences, and no consistent attacker-profitability improvement over standard GHOST under the evaluated withhold-and-release adversary. The resulting picture is mixed but technically informative: explicit visibility changes the stability profile of subtree-based PoW consensus in a systematic way, but the reported operating point produces a clear trade-off rather than a uniform gain across all metrics.

Keywords: Blockchain · Proof-of-Work · fork-choice rule · GHOST · visibility asymmetry · consensus stability · mining fairness.

1 Introduction

Proof-of-Work (PoW) blockchains couple probabilistic leader election with a deterministic fork-choice rule over a growing block tree. Because blocks are mined concurrently and propagate with non-negligible delay, honest miners frequently observe short-lived forks. Bitcoin resolves such forks using the longest-chain rule

⋆ Corresponding author: abd.almsailem@ku.edu.kw

(LCR) [1], which chooses the chain with the greatest cumulative work. LCR is robust when the block interval is large relative to propagation delay, but its security and efficiency degrade as throughput is increased: stale blocks waste honest work and amplify incentives for strategic behavior [2–4].

Subtree-based rules such as GHOST (Greedy Heaviest-Observed SubTree) [5] partially recover stale-block work by selecting, at each branching point, the child with the heaviest observed subtree. This inclusive philosophy enables higher block rates (and inspired Ethereum's uncle mechanism [6]), but it assumes that honest miners have sufficiently similar views of the block tree. In realistic heterogeneous networks, propagation is heavy-tailed and topologically asymmetric since different miners observe different subsets of blocks at any time [2, 7, 3]. Under such *visibility asymmetry*, honest miners compute divergent subtree weights, which increases head oscillation and can worsen reward inequality. Moreover, an adversary can exploit the asymmetry by withholding blocks and releasing them strategically (stale-growth / timed-release attacks) to inflate apparent subtree weight and trigger reorganizations [8–10].

This paper proposes *Visibility-Aware GHOST* (VA-GHOST), a drop-in fork-choice rule that makes block visibility explicit. VA-GHOST estimates each block's network visibility from lightweight decentralized attestations (*visibility certificates*) embedded in subsequent blocks, assigns a visibility factor $\nu_b \in [0, 1]$, and computes subtree weight as the sum of visibility-weighted contributions. Newly revealed or poorly propagated blocks are downweighted until they become broadly observed, which reduces sensitivity to local-view differences while recovering standard GHOST behavior in near-synchronous regimes.

The empirical study reports results from an implementation that is aligned with the protocol semantics in Section 4: each node maintains a local block DAG, updates visibility counters on block receipt, embeds bounded visibility certificates in mined blocks, and recomputes its preferred chain by greedy VA-GHOST descent. These results are obtained from the node-local DAG implementation used throughout this study and therefore reflect the protocol behavior under the reference simulation model.

Contributions. The contributions of this work are fourfold.

- We formalize visibility asymmetry as a first-class source of subtree-weight divergence in PoW networks and relate it to reorganization dynamics, reward dispersion, and adversarial timed release.
- We define VA-GHOST as a subtree-based fork-choice rule with bounded visibility certificates, node-local visibility factors, and visibility-weighted subtree scoring.
- We implement the rule in a node-local DAG simulator derived from CBlockSim [11], including explicit certificate processing, bounded metadata, and adversary selection by target hash-power share.
- We provide an empirical evaluation showing that visibility-aware weighting consistently reduces reorganization frequency relative to LCR and standard GHOST, while also exposing a clear trade-off: reorganizations become less

frequent but deeper on average, fairness effects remain modest, and attacker-profitability gains over standard GHOST are not consistently observed under the adversary considered here.

Organization. Section 2 reviews related work. Section 3 defines the model and problem. Section 4 presents VA-GHOST. Sections 5 and 6 describe the evaluation and results, and Section 7 concludes.

2 Related Work

Propagation, forks, and incentives in PoW. Empirical measurements show that block propagation is heterogeneous and heavy-tailed, which increases stale rates and impacts mining incentives [2, 7]. Systematic analyses quantify how latency, block size, and interval affect security and throughput and can make strategic mining more attractive [3, 4]. Eclipse and routing attacks further highlight that adversaries can manipulate who sees which blocks and when [12, 9].

Inclusive fork choice and blockDAGs. GHOST [5] and related inclusive protocols [13, 6] credit stale blocks by using subtree weight (or explicit uncle rewards) to guide fork choice at high block rates. Several DAG-based protocols generalize this idea, using voting (SPECTRE) [14], clustering (PHANTOM/GhostDAG) [15], or total ordering (Conflux) [16] to scale throughput.

Attacks exploiting asynchrony and visibility. Stale-growth attacks exploit the fact that subtree-based rules count newly observed descendants immediately and fully; withholding and timed release can inflate apparent subtree weight and induce reorganizations [8]. Network-layer manipulation (e.g., routing attacks) can similarly create long-lived view discrepancies that destabilize fork choice [9, 10].

Positioning. Prior work largely treats block visibility as an implicit property of the network rather than an explicit consensus parameter [1–10, 12–16]. VA-GHOST differs by introducing a lightweight as well as a decentralized visibility-estimation signal embedded in blocks themselves and uses that signal to scale each block's contribution to subtree weight. This explicitly targets both (i) honest-view divergence under heterogeneous propagation and (ii) attack bursts based on strategic withholding.

3 Background

We briefly recall PoW fork formation and the GHOST fork-choice rule, then highlight how *visibility asymmetry* makes subtree weights diverge across honest miners.

3.1 PoW Forks and Local Views

In a PoW blockchain, blocks reference a parent to form a rooted tree with genesis G. Because blocks are mined concurrently and propagate with delay, honest miners may extend different tips temporarily, which creates forks. Each miner i maintains a local view $\mathcal{T}_i(t)$ that consists of the blocks it has received by time t. Let $\mathbb{I}_i(b,t) \in \{0,1\}$ indicate whether miner i has observed block b by time t. Due to heterogeneous latency and bandwidth, $\mathbb{I}_i(b,t)$ can differ across honest miners for non-trivial periods [2, 7, 3].

A useful idealized notion is the (time-dependent) *network visibility* of a block:

$$\nu_b(t) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}_i(b,t), \tag{1}$$

i.e., the fraction of miners that have received b by time t. In near-synchronous regimes $\nu_b(t)$ quickly approaches 1, but in realistic networks it can remain in $(0,1)$ for seconds (or longer), especially under churn, congestion, or adversarial withholding [9, 10].

3.2 GHOST and the Visibility-Asymmetry Problem

GHOST [5] selects the canonical chain by repeatedly choosing, at each branching point, the child with the largest *observed subtree weight*. In the simplest form, miner i weighs each locally known block b by computing

$$W_i(b,t) = \sum_{x \in \mathrm{Desc}(b)} \mathbb{I}_i(x,t), \tag{2}$$

then follows the path from genesis to the leaf obtained by greedily selecting the heaviest child at every fork.

When visibility is asymmetric, two honest miners can observe different descendant sets and thus obtain different $W_i(b,t)$ for the same b. This increases head oscillation and can worsen reward inequality because well-connected miners tend to see more descendants earlier, which hence makes their subtree estimates more "complete" and their blocks less likely to be displaced [3, 17]. More critically, strategic adversaries can exploit subtree sensitivity by withholding blocks and releasing them in bursts; this can temporarily inflate the observed subtree of the adversary's preferred branch, trigger reorganizations, and amplify its revenue (stale-growth/timed-release attacks) [8].

3.3 Problem Statement

Subtree-based fork choice conflates *local* visibility ("known to me") with *global* acceptance ("widely propagated"). The central question we address is:

Can we design a GHOST-style fork-choice rule that remains compatible with PoW mining yet explicitly accounts for block visibility, so that (i) honest miners' subtree weights become more consistent under heterogeneous propagation,

(ii) latency-driven reward bias is reduced, and (iii) timed-release attacks are dampened?

VA-GHOST answers this affirmatively by estimating visibility from on-chain attestations and using a visibility-weighted subtree metric.

4 Proposed Solution: Visibility-Aware GHOST (VA-GHOST)

In this section, we introduce *Visibility-Aware GHOST* (VA-GHOST), a fork-choice rule that augments the original GHOST protocol [5] with an explicit protocol-level notion of block visibility. VA-GHOST preserves the subtree-based philosophy of GHOST but replaces raw descendant counting with a visibility-weighted subtree metric derived from decentralized attestations embedded in blocks. The design goal is to make subtree evaluation reflect *globally disseminated* blocks rather than merely *locally observed* ones, which is intended to reduce subtree-weight inconsistency, mitigate connectivity-induced unfairness, and limit propagation-driven attack vectors such as stale-growth attacks [3, 8].

4.1 Design Objectives and Overview

VA-GHOST is designed to satisfy three primary objectives:

1. **Stability under heterogeneous propagation:** Subtree weights computed by honest nodes should be less sensitive to local visibility differences, so that fork-choice decisions converge even when network latency and connectivity are heterogeneous [2, 7, 3].
2. **Fairness and decentralization:** The protocol should reduce the structural reward premium enjoyed by well-connected miners whose advantage stems purely from lower latency to attenuate centralization pressure [3, 17].
3. **Robustness to propagation-based attacks:** The protocol should limit the effectiveness of strategies that exploit visibility asymmetry, such as stale-growth or timed-release attacks [8], as well as routing and partitioning attacks that induce long-lived view discrepancies [9, 10], while remaining compatible with standard PoW mining [1].

To achieve these objectives, VA-GHOST introduces two key mechanisms:

- A *visibility factor* $\nu_b \in [0, 1]$ for each block b, approximating how widely b has propagated across the network.
- A *visibility-weighted subtree metric* $W_i^{\mathrm{VA}}(b, t)$ that replaces GHOST's unweighted descendant count $W_i(b, t)$ as the fork-choice signal at node i and time t.

Visibility factors are inferred from *visibility certificates*, which are compact attestations embedded in newly mined blocks that summarize which blocks the miner considers sufficiently visible. As blocks and certificates propagate, nodes

update their local visibility estimates and apply the VA-GHOST rule to select the canonical chain. From the perspective of the existing PoW infrastructure, VA-GHOST modifies only the fork-choice rule and introduces a single additional header field. PoW, transaction selection, and network transport remain unchanged and can benefit from existing relay optimizations such as FIBRE and compact blocks [18, 2].

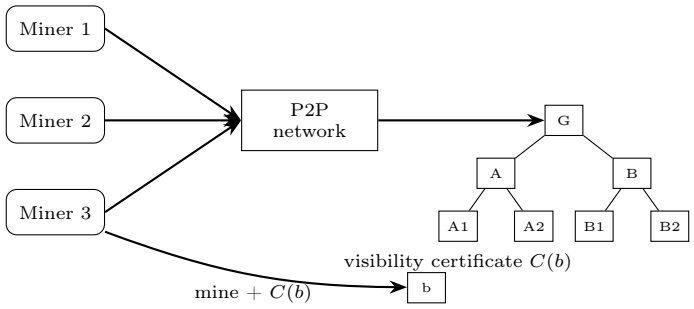


Fig. 1. High-level overview of VA-GHOST. Miners select parents using a visibility-weighted fork-choice rule, embed visibility certificates in newly mined blocks, and update local visibility estimates upon reception of blocks and certificates.

4.2 Visibility Certificates and Local Counters

Consider a PoW network with miner set $\mathcal{M} = \{1, \ldots, n\}$ and block set $B(t)$ observed by time t. Each miner $i \in \mathcal{M}$ maintains:

- A local block tree (or DAG) $\mathcal{T}_i(t)$, rooted at the genesis block.
- A *visibility counter* $k_b^{(i)}(t) \in \mathbb{N}$ for each block $b \in B(t)$ it is aware of.

The counter $k_b^{(i)}(t)$ is miner i's local estimate of how many attestations of b it has observed (including its own) and serves as a proxy for the unknown global visibility $\nu_b(t)$ studied in Section 3.

When miner i mines a new block b_{new} at time t, it constructs a *visibility certificate*

$$C(b_{\text{new}}) \subseteq B(t), \tag{3}$$

consisting of identifiers of blocks that i deems "sufficiently visible" at that time. A simple rule is to include all blocks b such that

$$k_b^{(i)}(t) \geq \tau_v, \tag{4}$$

where $\tau_v \in \mathbb{N}$ is a visibility threshold parameter. In practice, the candidate set $\{b \in B(t) : k_b^{(i)}(t) \geq \tau_v\}$ may exceed the budget $L_{\max}$, so the miner applies a

publicly specified reduction rule to output at most $L_{\max}$ entries. A simple deterministic policy is to sort candidates by a canonical key (e.g., decreasing height, then lexicographic block identifier) and keep the first $L_{\max}$. Alternatively, the miner can sample pseudorandomly using a seed derived from public block-header material (e.g., $H(p\|\text{nonce})$), so that—given the same candidate set—any verifier can recompute the same subset. More sophisticated policies can incorporate application-layer priorities (recency, fee weight, or ancestry contention) as long as the rule is fixed and does not permit strategic cherry-picking. VA-GHOST is agnostic to the specific policy provided it is deterministic or pseudorandom and any randomness is publicly derivable.

The certificate $C(b_{\text{new}})$ is serialized and embedded into a dedicated header field of b_{new}. The miner then broadcasts b_{new} as in conventional PoW systems [1, 6]. When another miner j receives b_{new} and validates its PoW and structure, it performs the following visibility updates:

1. It increments the counter for b_{new} itself,
$$k^{(j)}_{b_{\text{new}}} \leftarrow k^{(j)}_{b_{\text{new}}} + 1, \tag{5}$$
and this reflects that b_{new} has reached an additional node.
2. For each block identifier $x \in C(b_{\text{new}})$ that j recognizes in its local tree $\mathcal{T}_j(t)$, it increments the corresponding counter,
$$k^{(j)}_{x} \leftarrow k^{(j)}_{x} + 1. \tag{6}$$

Thus, each certificate functions as a vector of attestations of the form "I, a miner, have observed these blocks." As blocks propagate and new certificates accumulate, the counters $\{k^{(i)}_b\}$ at honest miners grow roughly in proportion to the number of distinct mining events that reference each block, similar in spirit to how blockDAG protocols infer global ordering from local edge observations [14–16].

Each miner then maps local counters to *visibility factors* via a monotone normalization function:
$$\nu^{(i)}_b(t) = f_{\text{vis}}\big(k^{(i)}_b(t)\big), \tag{7}$$
where $f_{\text{vis}} : \mathbb{N} \to [0, 1]$ is non-decreasing. A natural choice is the saturating linear function:
$$\nu^{(i)}_b(t) = \min\left(1, \frac{k^{(i)}_b(t)}{\tau_v}\right), \tag{8}$$
which interprets τ_v distinct attestations as sufficient evidence that b is widely disseminated. Other choices include concave functions that discount early attestations or functions with a hard cutoff that assign $\nu^{(i)}_b(t) = 0$ below a minimum number of attestations.

This mechanism is fully decentralized: no special nodes or trusted infrastructure are required to track visibility. Each miner updates its local counters based only on the certificates it receives, analogous to how blocks are already

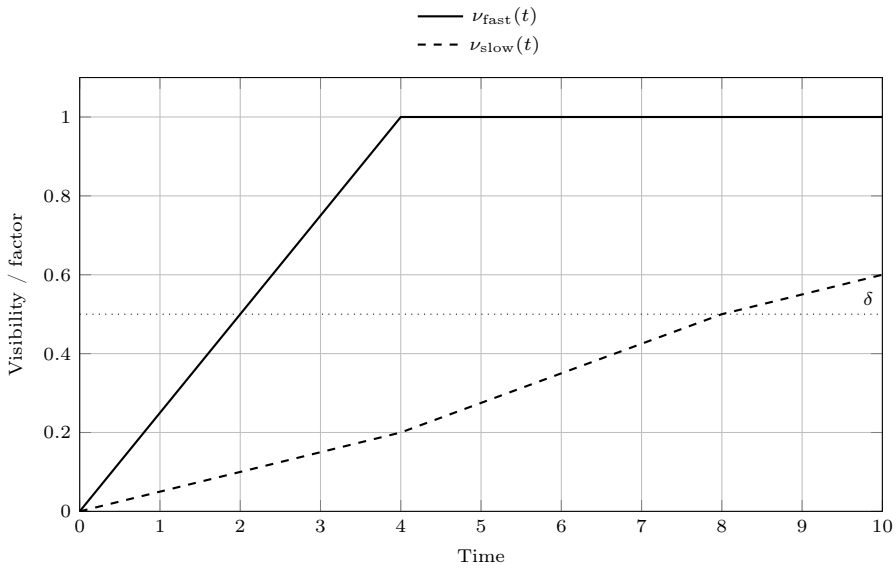


Fig. 2. Illustrative evolution of visibility counters $k_b^{(i)}(t)$ and derived visibility factors $\nu_b^{(i)}(t)$ as blocks and certificates are propagated. Blocks frequently referenced in certificates quickly reach $\nu_b^{(i)} \approx 1$, whereas slowly propagated or withheld blocks remain at low visibility for longer.

processed in today's P2P networks [2]. The bandwidth overhead is controlled by the certificate size bound $L_{\max}$, and the state overhead is a constant-size counter and factor per known block. All in all, VA-GHOST augments each block with a visibility certificate and requires nodes to maintain per-block visibility metadata, but these additions do not change the asymptotic space complexity of the protocol. In a standard PoW + GHOST implementation, a full node already stores the block tree (or DAG), headers, and transaction payloads, which together require $O(B)$ space in the number of blocks B. VA-GHOST adds only (i) a small per-block counter and derived visibility factor in local state, and (ii) a bounded-size list of block identifiers in each certificate field, whose maximum length $L_{\max}$ is a protocol constant. Consequently, the total storage at a full node remains $O(B)$, with a modest constant-factor increase in header and in-memory metadata size. In practice, implementations can further garbage-collect visibility metadata for blocks that lie far beyond the reorganization horizon, so the steady-state memory footprint is dominated by the existing blockchain data structure rather than by visibility information.

4.3 Visibility-Weighted Subtree Metric

Given visibility factors, VA-GHOST modifies GHOST's subtree metric to attenuate the influence of blocks that have low estimated visibility. Let $\mathcal{T}_i(t)$ denote

miner i's local block tree at time t, and let $\mathcal{D}_i(b,t)$ denote the set of descendants of block b in $\mathcal{T}_i(t)$. Standard GHOST uses the unweighted subtree size

$$W_i(b,t) = |\mathcal{D}_i(b,t)| + 1, \tag{9}$$

and selects the chain head via the greedy descent rule, and recursively chooses children that maximize W_i [5]. In VA-GHOST, we define a *visibility-weighted* subtree metric

$$W_i^{\mathrm{VA}}(b,t) = 1 + \sum_{d \in \mathcal{D}_i(b,t)} g\big(\nu_d^{(i)}(t)\big), \tag{10}$$

where $g : [0,1] \to \mathbb{R}_{\geq 0}$ is a weighting function that determines how strongly descendants with a given visibility factor contribute to subtree weight.

A simple and interpretable choice is a two-parameter piecewise linear function parameterized by a visibility threshold $\delta \in (0,1]$ and a discount factor $\alpha \in (0,1]$:

$$g(\nu) = \begin{cases} \alpha\nu, & 0 \leq \nu < \delta, \\ \nu, & \delta \leq \nu \leq 1. \end{cases} \tag{11}$$

Under this scheme, descendants with visibility below δ contribute a discounted weight $\alpha\nu$, whereas descendants with visibility at least δ contribute weight proportional to ν. If $\alpha < 1$, low-visibility blocks have strictly less influence than they would under standard GHOST, which corresponds to $g(\nu) \equiv 1$ for all locally visible blocks (implicitly assuming $\nu = 1$).

The VA-GHOST fork-choice rule mirrors GHOST's greedy traversal but uses W_i^{VA} instead of W_i. Starting from the genesis block g, miner i constructs a path $(b_0, b_1, \ldots, b_k)$ such that $b_0 = g$ and

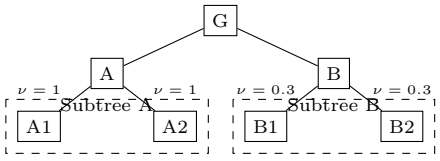
$$b_{j+1} = \arg \max_{c \in \mathrm{children}(b_j)} W_i^{\mathrm{VA}}(c,t), \tag{12}$$

for each level j. The local chain head at time t is then

$$\mathrm{head}_{\mathrm{VA}}^{(i)}(t) = b_k. \tag{13}$$

Intuitively, VA-GHOST treats descendants that have accumulated many attestations (i.e., large $\nu_d^{(i)}$) as strong evidence of honest work and allows them to contribute nearly their full weight, while descendants that are visible only to a small subset of miners contribute only fractionally until they propagate more widely. So, blocks that are newly created, slowly propagated, or strategically withheld have reduced leverage for reweighting subtrees; and this addresses the visibility asymmetry highlighted in Section 3.

In the limit where $\nu_d^{(i)}(t) \equiv 1$ for all blocks and $g(\nu) \equiv 1$, $W_i^{\mathrm{VA}}(b,t)$ coincides with $W_i(b,t)$ and VA-GHOST reduces to GHOST. In highly synchronous regimes where propagation is fast and homogeneous [2, 3], visibility factors quickly saturate at 1, and VA-GHOST behaves nearly identically to GHOST. In heterogeneous or adversarial regimes, as explained, VA-GHOST smooths subtree weights and reduces the influence of local propagation luck and withheld blocks.



	GHOST weight	VA-GHOST weight
Subtree A	$1 + 2 = 3$	$1 + 1 + 1 = 3$
Subtree B	$1 + 2 = 3$	$1 + 0.15 + 0.15 \approx 1.3$

Fig. 3. Conceptual comparison of subtree weights under GHOST and VA-GHOST. In the depicted fork, the two branches have the same raw descendant count, but descendants under one branch have low visibility. Standard GHOST assigns equal weight to both branches, while VA-GHOST discounts low-visibility descendants, which thus favors the branch with higher global visibility.

4.4 Node-Side VA-GHOST Logic

To make the operation of VA-GHOST precise, we now formalize the local logic executed by each miner. Algorithm 1 specifies the node-side behavior on the *receive path*: OnReceiveBlock processes incoming blocks and their visibility certificates, updates local visibility state, and recomputes the VA-GHOST head. Algorithm 2 specifies the *mining path*: OnMiningEvent constructs and mines new blocks on the current VA-GHOST head using visibility-aware certificates. Each node maintains a local block tree $\mathcal{T}_i$, per-block visibility counters $k_b^{(i)}$, derived visibility factors $\nu_b^{(i)} = f_{\text{vis}}(k_b^{(i)})$, and a visibility-weighted subtree metric $W_i^{\text{VA}}(b)$ that is updated incrementally whenever attestations are observed. The fork-choice rule is implemented by a GHOST-style greedy descent from genesis, where at each fork the child with the largest visibility-weighted subtree is selected. Unlike classical GHOST, which counts all locally visible descendants with unit weight, VA-GHOST modulates each block's contribution by $g(\nu_b^{(i)})$ and this therefore reduces the influence of blocks that have not yet propagated widely.

Implementation view. Figure 4 summarizes the node-side control flow: received blocks update visibility counters and subtree aggregates before head recomputation, while mining events build a new block on the current head with a fresh certificate.

4.5 Complexity, Overhead, and Compatibility

In terms of asymptotic complexity, VA-GHOST preserves the behavior of GHOST. For both protocols, the dominant cost is maintaining subtree aggregates and traversing the tree to identify the chain head. A naive implementation requires $O(|\mathcal{T}_i(t)|)$ time per head recomputation, but in practice incremental maintenance reduces this cost substantially. VA-GHOST adds $O(|C(b)|)$ work per received block to process its certificate and update counters; if $|C(b)|$ is bounded by a constant $L_{\max}$, this is a constant-factor overhead.

Algorithm 1 VA-GHOST receive and visibility update at miner i

Part I: Receive / counter update

State at node i:
$\mathcal{T}_i = (B_i, E_i)$: local block tree (blocks and parent edges)
$k_b^{(i)} \in \mathbb{N}$: visibility counter for each $b \in B_i$
$\nu_b^{(i)} = f_{\text{vis}}(k_b^{(i)}) \in [0,1]$: visibility factor
$g_b^{(i)} = g(\nu_b^{(i)}) \geq 0$: per-block contribution
$W_i^{\text{VA}}(b)$: visibility-weighted subtree metric
$p_i(b)$: parent of b in $\mathcal{T}_i$
$\text{Ch}_i(b)$: children of b in $\mathcal{T}_i$
$h_i = \text{head}_{\text{VA}}^{(i)}$: current local chain head

procedure OnReceiveBlock$(b, C(b))$
  Verify PoW, header, and parent link of b
  if $b \notin B_i$ **then** ▷ First time seeing b
    Insert b and $p_i(b)$ into $\mathcal{T}_i$; add edge $(p_i(b), b)$
    Initialize $k_b^{(i)} \leftarrow 0$, $\nu_b^{(i)} \leftarrow 0$, $g_b^{(i)} \leftarrow 0$
    Initialize $W_i^{\text{VA}}(b) \leftarrow 0$
  end if
  $k_b^{(i)} \leftarrow k_b^{(i)} + 1$
  UpdateVisibility(b)
  for all $x \in C(b)$ **such that** $x \in B_i$ **do**
    $k_x^{(i)} \leftarrow k_x^{(i)} + 1$
    UpdateVisibility(x)
  end for
  RecomputeHead
end procedure

Part II: Visibility recomputation helpers

function UpdateVisibility(x)
  $\nu_{\text{old}} \leftarrow \nu_x^{(i)}$
  $g_{\text{old}} \leftarrow g_x^{(i)}$
  $\nu_x^{(i)} \leftarrow f_{\text{vis}}(k_x^{(i)})$
  $g_x^{(i)} \leftarrow g(\nu_x^{(i)})$
  $\Delta \leftarrow g_x^{(i)} - g_{\text{old}}$
  if $\Delta = 0$ **then**
    return
  end if ▷ Propagate change to subtree weights along ancestor chain
  $v \leftarrow x$
  while v **is defined do**
    $W_i^{\text{VA}}(v) \leftarrow W_i^{\text{VA}}(v) + \Delta$
    $v \leftarrow p_i(v)$
  end while
end function
function RecomputeHead
  $b \leftarrow$ genesis block in $\mathcal{T}_i$
  while $\text{Ch}_i(b) \neq \emptyset$ **do**
    $b^\star \leftarrow b$
    $w^\star \leftarrow -\infty$
    for all $c \in \text{Ch}_i(b)$ **do**
      $w \leftarrow W_i^{\text{VA}}(c)$
      if $w > w^\star$ **then**
        $w^\star \leftarrow w$, $b^\star \leftarrow c$
      end if
    end for
    $b \leftarrow b^\star$
  end while
  $h_i \leftarrow b$
end function

Storage overhead is similarly modest; as full nodes (as per GHOST and Nakamoto's designs) already store the block tree, transaction data, and header metadata for each block [1, 6]. VA-GHOST adds two small per-block fields at each node: a counter $k_b^{(i)}$ and a visibility factor $\nu_b^{(i)}$. These can be stored as integer and floating-point values, respectively, and represent $O(1)$ additional state per block. The visibility certificate field increases header size by at most $L_{\max}$ block identifiers per block. With $L_{\max} = 32$ and 32-byte block identifiers, the

Algorithm 2 VA-GHOST mining and certificate selection at miner i

procedure OnMiningEvent
  $p \leftarrow h_i$ ▷ Current parent is VA-GHOST head
  $C(b_{\text{new}}) \leftarrow$ SelectCertificateEntries$(k^{(i)}, \tau_v, L_{\max})$
  Assemble block b_{new} with parent p and certificate $C(b_{\text{new}})$
  Solve PoW for b_{new}
  Broadcast b_{new} on the P2P network
end procedure
function SelectCertificateEntries$(k^{(i)}, \tau_v, L_{\max})$
  $S \leftarrow \{b \in B_i : k_b^{(i)} \geq \tau_v\}$
  if $|S| > L_{\max}$ **then**
    Select a subset of $L_{\max}$ blocks from S (e.g., uniformly or
biased towards recent blocks)
  end if
  return S
end function

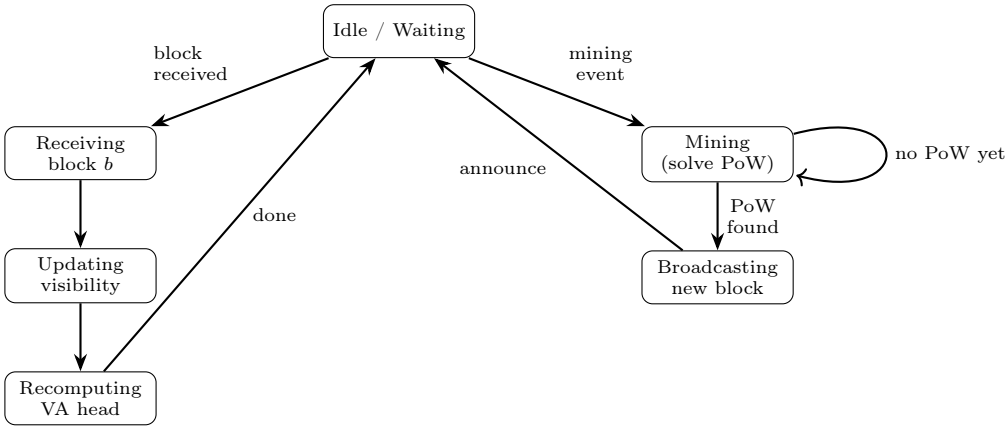


Fig. 4. High-level control flow of a VA-GHOST node.

resulting worst-case certificate payload is approximately 1 KB per block, which is a bounded and practically modest engineering overhead.

From a deployment perspective, VA-GHOST is backwards-compatible with existing PoW ecosystems in several respects:

- It *does not* change the PoW puzzle, difficulty adjustment, transaction format, or network topology.
- It can coexist with latency-reduction techniques such as FIBRE [18] and other relay networks, which in fact improve the convergence of visibility factors by accelerating block and certificate dissemination.
- In regimes where propagation is fast and homogeneous, VA-GHOST approximates GHOST, enabling incremental adoption and comparative evaluation.

Conceptually, VA-GHOST reinterprets subtree-based fork-choice in light of the visibility asymmetry documented in prior work [2, 7, 3, 9, 8]. Rather than assuming that local visibility implies global acceptance, it embeds a lightweight visibility estimation mechanism into the protocol and uses visibility-weighted subtree metrics as the fork-choice signal.

Remark (Tamper-resistance and misreporting of visibility). Since the visibility certificate $C(b)$ is embedded in the block header and thus covered by the PoW hash of block b, modifying the certificate of an already-mined block without recomputing PoW is infeasible under the standard hash-function assumptions. Any attempt to tamper with $C(b)$ in transit changes the block header, invalidates the original block hash, and is therefore rejected by honest nodes. So, adversaries cannot retroactively alter the visibility attestations of blocks they did not mine; they can only choose the contents of certificates for blocks they themselves produce.

A malicious miner may, however, misreport its own view by including in $C(b)$ identifiers of blocks it has not actually received. In VA-GHOST, such misreporting cannot amplify visibility "for free" because each additional attestation of a block x must be carried in a valid PoW block, and honest miners simultaneously contribute attestations to genuinely well-propagated blocks. Moreover, the choice of the normalization function f_{vis} and weighting function $g(\cdot)$, together with the threshold parameter τ_v and the certificate size bound $L_{\max}$, bounds the marginal influence of low-visibility blocks on the visibility-weighted subtree metric $W_i^{\text{VA}}(\cdot)$. Intuitively, inflating the visibility factor ν_x of a block beyond what is justified by real network propagation requires proportional computational effort (hashrate) expenditure, and low-visibility blocks remain discounted until they accumulate a sufficient number of independent attestations.

Remark (Liveness and parameterization). The introduction of visibility factors raises a natural concern about liveness: if new blocks initially have low visibility ν_b and are therefore heavily discounted, VA-GHOST could in principle slow down or destabilize tip selection. In practice, liveness is preserved for two reasons. First, every newly mined block b always contributes at least its own unit weight, via the "+1" term in (10), and thus receives an immediate self-attestation when it is created, so its visibility factor ν_b does not remain at zero. Second, as blocks and certificates propagate, honest miners quickly accumulate attestations for recently produced blocks, and this causes ν_b to approach 1 and $g(\nu_b)$ to converge to the unit contribution of classical GHOST.

The parameters f_{vis}, $g(\cdot)$, τ_v, and $L_{\max}$ therefore act as a tunable interface between conservativeness and responsiveness. Larger values of τ_v and more aggressively concave choices of $g(\cdot)$ slow down the rate at which blocks reach full weight, and this yields stronger damping of low-visibility bursts at the cost of slightly slower reaction to legitimate forks. But, smaller values of τ_v and more linear $g(\cdot)$ recover behavior closer to standard GHOST, with minimal impact on throughput in near-synchronous regimes. In the idealized limit where $\nu_b \equiv 1$ for all blocks and $g(\nu) \equiv 1$, the visibility-weighted metric $W_i^{\text{VA}}(b)$ coincides

with the unweighted subtree size $W_i(b)$, and VA-GHOST reduces exactly to GHOST. Thus, for reasonable parameter choices, VA-GHOST inherits the liveness properties of GHOST in well-connected networks and is designed to provide additional robustness in heterogeneous and adversarial settings, although the empirical strength of that robustness remains a question for evaluation.

Observation 1 (Reduction to GHOST). If $\nu_b^{(i)}(t) = 1$ for all relevant blocks and $g(\nu) \equiv 1$, then $W_i^{\mathrm{VA}}(b,t)$ reduces to the standard GHOST subtree size, and VA-GHOST selects the same preferred chain as GHOST.

Observation 2 (Bounded certificate overhead). Each certificate contains at most $L_{\max}$ block identifiers. In the implementation used for evaluation, $L_{\max} = 32$; assuming 32-byte block identifiers, the worst-case certificate payload is approximately $32 \times 32 = 1024$ bytes per block. This is a bounded engineering overhead rather than an overhead that grows with the chain size or network size.

Observation 3 (Rate-limited certificate inflation). A colluding coalition can only increase visibility counters through certificates carried in adversarially mined blocks. Since each block can carry at most $L_{\max}$ references, any attempted inflation of low-visibility branches is rate-limited by adversarial block production and by the bounded certificate size. We do *not* claim that this eliminates stronger collusive certificate-inflation strategies; rather, it bounds the per-block influence of any single adversarial release and clarifies the stronger attack variant that remains outside the present evaluation scope.

The next section describes the experimental setup used to evaluate VA-GHOST against the foundational PoW rule (LCR) and standard GHOST in terms of stability, fairness, and robustness to propagation-based attacks.

5 Experimental Setup

We evaluate VA-GHOST against LCR and standard GHOST using a discrete-event PoW network simulator derived from CBlockSim [11] (which is inspired by SimBlock [19]). All reported numbers are obtained from a node-local DAG implementation of VA-GHOST: each node stores its locally known blocks and parent/child relationships, maintains visibility counters, embeds a bounded certificate field in newly mined blocks, and recomputes its preferred chain by greedy VA-GHOST descent after block receipt. This implementation operationalizes the protocol mechanics described in Section 4 and serves as the reference implementation used throughout the empirical study.

For each configuration we run three independent trials with different random seeds and report sample means.

5.1 Baseline Network, Protocol, and Implementation Parameters

Unless stated otherwise, experiments use $N = 1000$ miners across $R = 6$ geographic regions with a small-world overlay topology (average degree $\overline{d} = 80$).

Blocks are mined with target interval $T_{\text{blk}} = 2\,\text{s}$ and incur a fixed validation time $T_{\text{val}} = 0.34\,\text{s}$. Block payload size is 0.18 Mb. The transaction/data plane is disabled so the evaluation isolates block-level consensus behavior. Table 1 summarizes the shared network/workload configuration, while Table 2 lists the fixed VA-GHOST parameters used in *all* scenarios. The same parameterization is used in all scenarios; parameters are not retuned per scenario.

Table 1. Shared simulator configuration.

Parameter	**Symbol**	**Value**
Number of nodes	N	1000
Regions	R	6
Avg. degree	$\overline{d}$	80
Target block interval	T_{blk}	2 s
Simulation time	T_{sim}	6000 s
Block size (payload)	B_{size}	0.18 Mb
Block validation time	T_{val}	0.34 s
Block reward	R_{blk}	12.5 units
Uncle reward	R_{uncle}	$R_{\text{blk}}/32$

Table 2. Fixed VA-GHOST parameters used across all scenarios.

Parameter	**Symbol**	**Value**
Visibility threshold	τ_v	4
Certificate size bound	$L_{\max}$	32
Visibility break-point	δ	0.5
Low-visibility discount	α	0.5

The practical metadata overhead is easy to state concretely. With $L_{\max} = 32$ and 32-byte block identifiers, the worst-case certificate payload is approximately 1024 bytes per block. This is bounded and independent of the chain length and network size.

5.2 Bounded Sensitivity Analysis

A full multi-parameter sweep is left to subsequent work, but the fixed-parameter design in Table 2 still admits a useful *bounded sensitivity analysis*. This subsection does not attempt to identify globally optimal settings; instead, it characterizes how the main VA-GHOST parameters are expected to affect behavior locally around the reported operating point.

At a high level, the parameters control four distinct levers: how much evidence is needed before a block is treated as well disseminated (τ_v), how much certificate metadata can be attached to a block ($L_{\max}$), where the discount regime transitions from harsh to neutral (δ), and how harsh the low-visibility discount is below that transition (α). Their directional effects are summarized in Table 3. These directions follow directly from the protocol equations and the implementation: raising τ_v makes visibility harder to accumulate, raising $L_{\max}$ allows more visibility evidence to be propagated per mined block, raising δ expands the regime in which descendants are discounted, and lowering α makes that discount sharper.

Table 3. Bounded directional sensitivity of the fixed VA-GHOST parameters around the reported operating point.

Parameter	**Change**	**Expected directional effect near current operating point**
τ_v	increase	More conservative visibility recognition; fewer descendants reach full weight quickly; likely stronger suppression of locally visible but weakly disseminated subtrees, at the cost of slower recovery of honest subtree mass.
$L_{\max}$	increase	More visibility references can be carried per block; faster dissemination of visibility evidence; reduced certificate truncation bias, with linearly higher metadata overhead.
δ	increase	Larger portion of the visibility range remains in the discounted regime; tends to strengthen skepticism toward partially visible descendants.
α	decrease	Harsher penalty for low-visibility descendants; can further damp local head churn, but may also deepen corrections when visibility eventually catches up.

The present empirical pattern — substantially fewer reorganizations but deeper average reorganizations — is consistent with this bounded sensitivity picture. It suggests that the current fixed setting ($\tau_v = 4,\ L_{\max} = 32,\ \delta = 0.5,\ \alpha = 0.5$) places the reported operating point in a *conservative visibility-weighting regime*: subtree growth on weakly disseminated branches is discounted enough to reduce head churn, but once enough visibility evidence accumulates, corrections can arrive in larger steps. This interpretation strengthens the paper's main claim without overreaching: the implementation reveals a meaningful fork-choice trade-off, while a full retuning study is left to subsequent work rather than folded implicitly into the reported operating point.

5.3 Adversary Model and Scope

We evaluate a *withhold-and-release coalition* adversary. The adversarial coalition is selected to approximate a target *hash-power share* (rather than a raw node

fraction), privately shares newly mined blocks within the coalition, and withholds them from honest miners. Release is triggered by one of three conditions: a maximum hold time, a sufficient private lead, or an impending catch-up of the public chain. This model captures a concrete stale-growth / timed-release family of attacks, but it is intentionally narrower than full real-world adversarial behavior.

In particular, the present evaluation does *not* model BGP hijacks, eclipse attacks, dynamic routing changes, or an adversary that can coordinate arbitrary out-of-band cross-attestations beyond what it can carry in the certificate field of blocks it mines. We therefore interpret the results strictly within the evaluated conditions.

5.4 Evaluation Scenarios

We study four scenarios designed to isolate throughput/stability, fairness under propagation heterogeneity, and robustness to withholding attacks:

- **Scenario A (baseline, no attacker):** heterogeneous (Gaussian) hash power and the default region distribution; attacker disabled.
- **Scenario B (fairness stress-test):** uniform hash power for all miners and a skewed region distribution to amplify propagation asymmetry; attacker disabled.
- **Scenario C (moderate attacker):** a withhold-and-release coalition targeting approximately 10% total hash power with a 5 s withholding horizon.
- **Scenario D (strong attacker):** a withhold-and-release coalition targeting approximately 30% total hash power with a 10 s withholding horizon.

5.5 Metrics

We report: (i) **stale/uncle rate**, (ii) **reorganization depth** and **reorg count**, and (iii) **block propagation delay (BPD)** percentiles measured from block creation to reception. For fairness, we compute each miner's reward share r_i and hash-power share p_i, and the ratio

$$\gamma_i = \frac{r_i}{p_i}, \tag{14}$$

where $\gamma_i \approx 1$ indicates proportional rewards. We summarize fairness using $\mathrm{Std}(\gamma_i)$ and the Gini coefficient of rewards.

For attack scenarios, we partition miners into attacker set $\mathcal{A}$ and honest set $\mathcal{H}$ and report group-level reward/hash multipliers:

$$\Gamma_{\mathcal{A}} = \frac{\sum_{i\in\mathcal{A}} r_i}{\sum_{i\in\mathcal{A}} p_i}, \qquad \Gamma_{\mathcal{H}} = \frac{\sum_{i\in\mathcal{H}} r_i}{\sum_{i\in\mathcal{H}} p_i}. \tag{15}$$

An attack is profitable when $\Gamma_{\mathcal{A}} > 1$.

Finally, we emphasize that *reorg count* is measured as the aggregate number of reorganization events observed across nodes over the entire simulation, so the absolute values are large; they are intended for *comparative* interpretation across policies rather than as standalone absolute rates.

6 Results and Discussion

We summarize the main quantitative findings across Scenarios A–D. Unless stated otherwise, values are sample means over three runs.

6.1 Scenario A: Baseline Non-Adversarial Network

Table 4 reports consensus-level metrics under the baseline heterogeneous but non-adversarial network. The most consistent effect of VA-GHOST is a large reduction in reorganization count: relative to standard GHOST, the aggregate reorg count drops from approximately 308,070 to 174,052. However, the average reorganization depth increases substantially from about 0.65 to 1.06. The stale/uncle rate remains between LCR and GHOST, and the 50%/90% BPD values remain essentially unchanged across policies. Fairness metrics are also close across policies. The clearest baseline conclusion is therefore not uniform improvement across all stability metrics, but a distinct trade-off: *fewer reorgs at the cost of deeper reorgs*.

Figure 5 visualizes the same trade-off by showing the marked reduction in reorganization count together with the increase in average reorganization depth under VA-GHOST.

Table 4. Scenario A: baseline heterogeneous network (no attacker).

Policy	Block time [s]	Stale/uncle rate	50% BPD [s]	90% BPD [s]	Avg reorg depth	Reorg count	Gini
LCR	1.980	0.204	0.373	1.311	0.638	302,790	0.367
GHOST	1.987	0.178	0.373	1.308	0.648	308,070	0.363
VA-GHOST	1.939	0.196	0.373	1.309	1.059	174,052	0.364

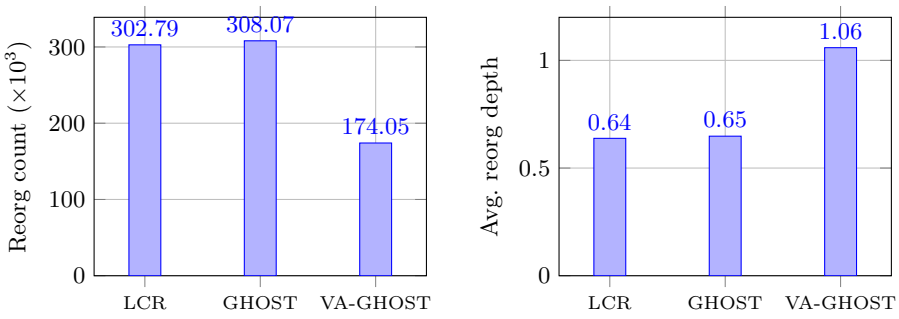


Fig. 5. Scenario A visualization. VA-GHOST substantially reduces aggregate reorganization count, but increases average reorganization depth.

6.2 Scenario B: Fairness Under Heterogeneous Propagation

Scenario B isolates connectivity-induced bias by giving all miners equal hash power but skewing the region distribution. Because every miner has the same p_i, the reward–hash correlation is not especially informative here; the analysis therefore focuses on dispersion and inequality. Table 5 shows that VA-GHOST again sharply reduces reorg count relative to both LCR and GHOST, but it also exhibits the deepest reorganizations. On fairness, the picture is mixed and modest rather than dramatic. The standard deviation of γ_i and the reward Gini are slightly better than LCR but not better than GHOST. Accordingly, the reported results do *not* support a strong claim that VA-GHOST dominates standard GHOST on fairness in this scenario.

Figure 6 summarizes the corresponding stability and fairness indicators, highlighting that the main effect in this setting is reduced reorganization count rather than a strong fairness advantage over standard GHOST.

Table 5. Scenario B: fairness stress-test (uniform hash power, heterogeneous propagation).

Policy	Std(γ_i)	**Gini**	**Avg reorg depth**	**Reorg count**	**Stale/uncle rate**
LCR	0.648	0.357	0.639	312,027	0.210
GHOST	0.626	0.345	0.678	337,560	0.190
VA-GHOST	0.634	0.348	1.077	182,906	0.206

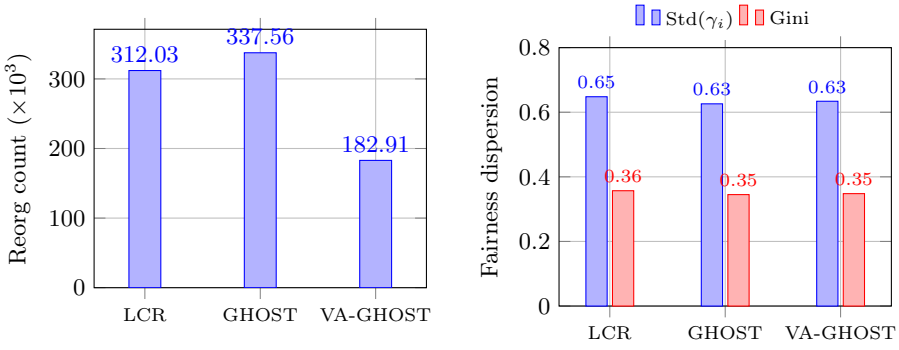


Fig. 6. Scenario B visualization. VA-GHOST sharply reduces reorganization count, while fairness gains remain modest and do not clearly dominate standard GHOST.

6.3 Scenario C: Moderate Withhold-and-Release Attacker

Scenario C introduces a coalition with target adversarial hash-power share of approximately 10% and a 5 s withholding horizon. Table 6 again shows the same stability trade-off: VA-GHOST reduces aggregate reorg count by roughly 45% relative to standard GHOST, but average reorg depth increases from about 0.64 to 1.06. The attacker multiplier $\Gamma_{\mathcal{A}}$ is close to 1 for both GHOST and VA-GHOST, with VA-GHOST very slightly higher in the present runs (0.997 vs. 0.991). Consequently, the implementation does not support a strong claim that VA-GHOST improves attacker profitability over standard GHOST in this moderate setting. The more defensible statement is that VA-GHOST changes the stability profile under attack, but the incentive effect is limited under the current parameterization.

Figure 7 presents the moderate-attack results in visual form, showing both the reduction in reorganization count and the near-proportional attacker reward multiplier under the current parameterization.

Table 6. Scenario C: moderate withhold-and-release attacker.

Policy	Stale/uncle rate	Avg reorg depth	Reorg count	$\Gamma_{\mathcal{A}}$	$\Gamma_{\mathcal{H}}$	Gini
LCR	0.201	0.646	301,318	0.947	1.006	0.366
GHOST	0.178	0.643	314,070	0.991	1.001	0.367
VA-GHOST	0.198	1.055	171,514	0.997	1.000	0.378

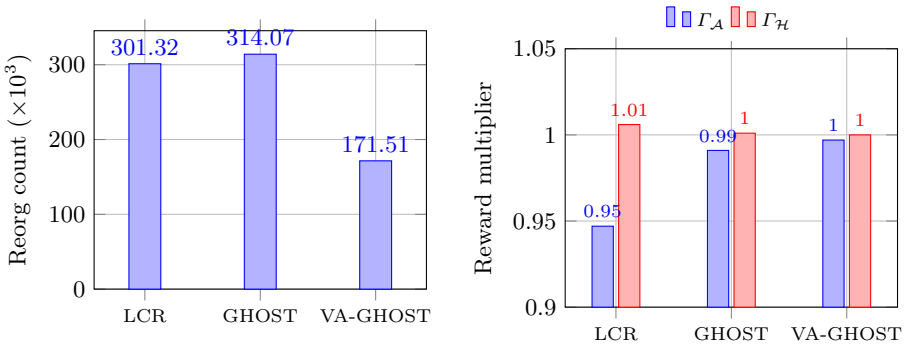


Fig. 7. Scenario C visualization. VA-GHOST reduces reorganization count substantially, but does not show a clear attacker-profitability advantage over standard GHOST in the moderate attack setting.

6.4 Scenario D: Strong Withhold-and-Release Attacker

Scenario D strengthens the adversary to approximately 30% target hash power and a 10 s withholding horizon. Table 7 shows that the overall pattern remains stable: VA-GHOST continues to reduce reorg count very substantially, but its average reorg depth remains much larger than that of LCR or standard GHOST. The attacker multiplier now falls slightly below 1 for both GHOST and VA-GHOST, with GHOST numerically lower (0.981 vs. 0.983). Again, the present data do not support a strong claim that VA-GHOST improves attack resilience relative to standard GHOST under this adversary; rather, both policies keep the attacker close to proportional while exhibiting different stability trade-offs.

Figure 8 shows the same comparison for the stronger attack setting, where VA-GHOST continues to reduce reorganization count but does not improve the attacker multiplier relative to standard GHOST.

Table 7. Scenario D: strong withhold-and-release attacker.

Policy	Stale/uncle rate	Avg reorg depth	Reorg count	$\Gamma_{\mathcal{A}}$	$\Gamma_{\mathcal{H}}$	Gini
LCR	0.201	0.646	309,831	1.025	0.989	0.374
GHOST	0.179	0.644	314,778	0.981	1.008	0.369
VA-GHOST	0.199	1.049	175,023	0.983	1.007	0.372

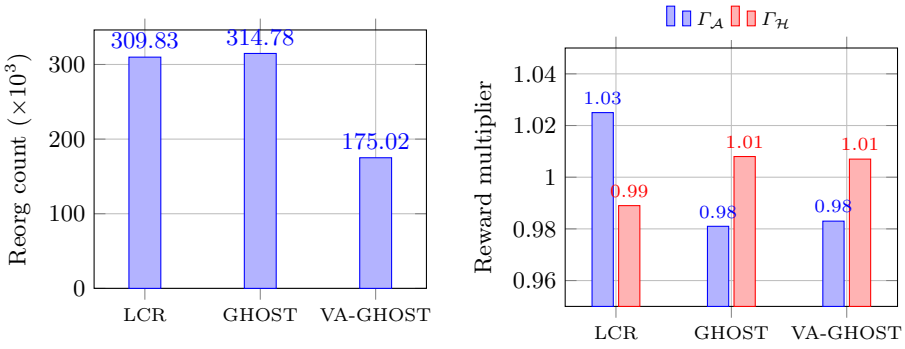


Fig. 8. Scenario D visualization. Under the stronger withhold-and-release setting, VA-GHOST continues to reduce reorganization count, while the attacker multiplier remains close to proportional and does not improve over standard GHOST.

6.5 Overall Findings Across Scenarios

Two conclusions are robust across all four scenarios. First, VA-GHOST consistently reduces aggregate reorganization count. Interpreted operationally, this

means that the preferred head changes less often, i.e., the fork-choice rule dampens frequent head churn. Second, this benefit is accompanied by a non-trivial trade-off: when a reorganization does occur, it is deeper on average than under LCR or standard GHOST. The fairness signal is modest and does not show a clean advantage over standard GHOST, while the attack scenarios do not show a consistent profitability benefit over standard GHOST under the reported implementation and parameters.

Taken together, these results do *not* support the strongest version of the original hypothesis (that VA-GHOST uniformly improves stability, fairness, and attacker resistance). They do, however, support a narrower and still meaningful claim: making visibility explicit changes the stability profile of subtree-based PoW consensus in a systematic way. In the reported implementation, the most stable empirical effect is reduced reorganization frequency; the main open design problem is how to preserve that benefit without paying for it with deeper reorganizations.

7 Conclusion

Visibility-Aware GHOST (VA-GHOST) augments subtree-based PoW fork choice with an explicit notion of block visibility, represented through lightweight bounded certificates and node-local visibility counters. The central question addressed in this study is not only whether visibility-awareness changes consensus behavior, but how it changes that behavior once the protocol is implemented as a node-local DAG rule rather than as a chain-level approximation.

The empirical results show that visibility-awareness has a systematic and non-trivial effect on the stability profile of the protocol. Across all four scenarios, VA-GHOST substantially reduces the number of reorganization events relative to both LCR and standard GHOST. This indicates that discounting weakly disseminated descendants can suppress frequent head oscillations and alter branch selection in a persistent way. At the same time, the same experiments show that reorganizations become deeper on average, that fairness gains are limited, and that the evaluated withhold-and-release adversary does not yield a consistent attacker-profitability advantage relative to standard GHOST. Taken together, these findings support a precise conclusion: visibility-awareness is a meaningful fork-choice lever, but under the parameterization studied here it introduces a clear frequency–depth trade-off rather than a uniform improvement along all axes.

Two broader implications follow. First, visibility should be treated as a first-class protocol signal in subtree-based PoW analysis, because local observation and broad dissemination are not equivalent in heterogeneous networks. Second, protocol evaluation must distinguish between conceptual promise and operating-point quality: the reported results show that the VA-GHOST mechanism is coherent and behaviorally significant, but they also indicate that the current certificate-selection rule and parameter setting do not yet shift the trade-off to an unequivocally favorable regime.

Several directions remain important. A larger sensitivity study over τ_v, $L_{\max}$, δ, and α is needed to determine whether the observed trade-off can be moved toward shallower reorganizations without sacrificing the reduction in reorganization frequency. In addition, richer adversary models, more realistic network traces, and alternative certificate-selection policies should be studied to test how robust the current conclusions are beyond the specific withhold-and-release setting used here.

References

1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
2. C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," in *Proc. IEEE 13th Int. Conf. Peer-to-Peer Computing (P2P)*, pp. 1–10, 2013.
3. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, pp. 3–16, 2016.
4. A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal Selfish Mining Strategies in Bitcoin," in *Proc. Int. Conf. Financial Cryptography and Data Security (FC)*, pp. 515–532, 2016.
5. Y. Sompolinsky and A. Zohar, "Secure High-Rate Transaction Processing in Bitcoin," in *Proc. Int. Conf. Financial Cryptography and Data Security (FC)*, pp. 507–527, 2015.
6. G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum Yellow Paper*, 2014.
7. J. A. Donet, C. Pérez-Solà, and J. Herrera-Joancomartí, "The Bitcoin P2P Network," in *Financial Cryptography and Data Security Workshops*, pp. 87–102, 2014.
8. H. Zhu, X. Chang, J. V. Mišić, V. B. Mišić, L. Han, and Z. Chen, "Is Stubborn Mining Severe in Imperfect GHOST Bitcoin-Like Blockchains? Quantitative Analysis," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2488–2501, 2024.
9. M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies," in *Proc. IEEE Symp. Security and Privacy (S&P)*, pp. 375–392, 2017.
10. C. Natoli and V. Gramoli, "The Balance Attack Against Proof-of-Work Blockchains: The R3 Testbed as an Example," *arXiv preprint* arXiv:1612.09426, 2016.
11. X. Ma, H. Wu, D. Xu, and K. Wolter, "CBlockSim: A Modular High-Performance Blockchain Simulator," in *Proc. IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC)*, pp. 1–5, 2022.
12. E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network," in *Proc. 24th USENIX Security Symp.*, pp. 129–144, 2015.
13. Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive Block Chain Protocols," in *Proc. Int. Conf. Financial Cryptography and Data Security (FC)*, pp. 528–547, 2015.
14. Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol," *IACR Cryptology ePrint Archive*, Report 2016/1159, 2016.

15. Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM and GHOSTDAG: A Scalable Generalization of Nakamoto Consensus," *IACR Cryptology ePrint Archive*, Report 2018/104, 2018.
16. C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. C.-C. Yao, "Scaling Nakamoto Consensus to Thousands of Transactions per Second," *CoRR*, abs/1805.03870, 2018.
17. Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, "Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 919–927, 2015.
18. M. Corallo, "FIBRE: Fast Internet Bitcoin Relay Engine," 2016. [Online]. Available: https://bitcoinfibre.org/
19. Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "SimBlock: A Blockchain Network Simulator," in *Proc. IEEE INFOCOM Workshops (CryBlock)*, pp. 325–329, 2019.