An Economic Framework for Trustless Decentralized Storage

May 2026

Abstract

Probabilistic proofs of storage give a verifier compact confirmation that a storage node holds an assigned dataset, without requiring full retrieval. In a network of untrusted storage nodes, such proofs close the verification gap. Verification alone, however, does not constitute enforcement: a rational storage node will defect whenever the expected gain exceeds the expected cost of detection, regardless of whether audits are in place.

We propose an economic framework that closes this incentive gap. The system pairs a continuous audit loop with a payment structure calibrated so that honest, low-latency local storage is strictly more profitable than any deviation: proxying data on demand, retaining only a subset of assigned blocks, or going silent. Each node accumulates a reputation derived from its audit history and response latency; each dataset accumulates a health score from the reputations of its assigned nodes. An assignment manager routes content toward high-reputation nodes and triggers automatic reassignment when health degrades.

We characterize the conditions under which honest storage is the dominant strategy under this mechanism, and examine the principal open questions: whether payment withholding alone suffices or whether explicit collateral commitment is warranted, the sybil attack surface introduced by a bounty-based peer verification model, and the extension of the framework to multi-provider erasure-coded deployments.

1 Introduction

Decentralized storage networks distribute data across untrusted, independent nodes. A client stores a dataset with a provider and receives an identifier by which the dataset can later be retrieved. For content to remain continuously available, it must be held by a persistent node, a storage provider, that retains it on the client's behalf. Storage providers currently operate entirely on trust. The client pays, specifies a dataset identifier, and assumes the provider is faithfully retaining every block of the stored dataset.

Cryptographic audit protocols [cite Ateniese et al., Juels-Kaliski, Shacham-Waters] close what we call the verification gap. These protocols enable a verifier to confirm with high probability that a remote storage node holds a complete dataset, by exchanging a compact challenge and proof rather than retrieving the data. A challenge of tens of bytes and a proof of hundreds of bytes can probabilistically audit a petabyte-scale archive. Applied to such networks, these protocols provide

a monitoring primitive: at any time, the operator can issue a challenge to a storage node and determine with measurable probability whether the node is holding its assigned content.

Decentralized storage networks make files accessible as long as some node holds them, but provide no mechanism to ensure that any particular node persists a file over time. The framework proposed here adds an economic layer for nodes that voluntarily accept long-term storage assignments. Nodes that decline to participate remain part of the network; the protocol is concerned only with nodes that take on the role of persistent storage provider. For those nodes, the audit loop and reputation system create an incentive structure that aligns the properties of a reliable storage provider, namely availability, fast retrieval, and persistent local retention, with the behaviors that build and sustain a node's reputation.

The economic layer closes this gap through the assignment mechanism. Nodes earn by holding assignments; assignments go to the highest-reputation nodes; reputation is derived continuously from audit performance. A node that stores content faithfully and responds promptly builds reputation and remains eligible for new assignments. A node that fails challenges, responds slowly, or goes silent loses reputation and is passed over in favor of better performers. Honest storage is simply the strategy that sustains a node's place in the assignment pool.

Response latency is the key distinguishing signal between honest storage and on-demand proxying. Local storage is fast; proxying challenged blocks from the network introduces additional latency in every audit round. The reputation score reflects the full spectrum of outcomes: fast responses are positive evidence of genuine local possession; slow responses are treated as evidence of proxying and penalize reputation accordingly. Non-response is treated as a liveness failure: a node that ceases responding triggers payment suspension and automatic reassignment of its content to healthier nodes. An invalid proof carries the strongest negative signal; a node that is reachable but cannot produce a valid response for the challenged blocks provides direct evidence that the assigned content is no longer intact. A payment function that decays with response time, combined with a reputation score that compounds over these outcomes, makes honest local storage the dominant strategy for every rational storage node.

The mechanism also addresses coordinated evasion. Because the operator controls storage assignments, nodes cannot select their co-assignees and therefore cannot arrange in advance to partition assigned content among colluders. A node that systematically declines new assignments is treated as suspicious, since this behavior is consistent with waiting for assignment opportunities that favor coordination, and repeated refusals contribute negatively to reputation. Correlated audit failures across co-assigned nodes provide an additional signal: nodes sharing a hidden failure domain or coordinating omissions exhibit temporally correlated failures, which the assignment manager uses to detect and break up suspected coalitions regardless of declared independence.

We formalize the interaction as a mechanism design problem in which the operator commits to a payment rule and reputation update procedure before nodes choose their strategies. This framing separates the enforcement question from the verification question: the soundness of the underlying audit protocol, which prevents a computationally bounded prover from forging valid proofs for absent content, is taken as given. The question this paper addresses is whether an economic layer, built on the audit loop, can make retaining content locally the profit-maximizing choice for every rational node.

2 The Storage Game

We formalize the interaction between the operator and storage nodes as a mechanism design problem. The operator is the mechanism designer: it commits to a set of rules, namely the payment function, the reputation update procedure, and the assignment algorithm, before storage nodes choose their strategies. Storage nodes are rational agents who observe the mechanism and select the strategy that maximizes their expected utility. The design goal is a mechanism under which honest storage is a dominant strategy for every storage node.

2.1 Players

The system involves four abstract audit roles, a reputation database, an assignment manager, and two classes of participants.

The audit cycle is defined by four roles: Tagger, Challenger, Prover, and Verifier. These are abstract interfaces; the actors performing each role may vary by deployment and protocol. The Tagger processes content at commitment time, computing a cryptographic tag for each block and assembling the result into a TagList. Depending on the protocol, the Tagger may be the client, generating metadata before upload, or a dedicated service. The Challenger issues periodic audit challenges to each active storage node; protocols requiring a trusted challenge authority use a centrally operated Challenger, while protocols supporting public verification permit any party to issue valid challenges. The Prover is always the storage node: it holds the assigned content and responds to audit challenges. The Verifier validates proof responses, measures response latency, and records outcomes in the reputation database.

The reputation database is the persistent state connecting audit outcomes to assignment decisions. The Verifier writes each audit outcome to this database; the reputation score of each storage node is derived from its recorded history of proof outcomes and response latencies.

The Assignment Manager is a process that, with input from the reputation database, generates the assignment mapping of datasets to storage nodes. Each dataset is assigned to k storage nodes, where k is a configurable replication factor chosen to meet the desired resilience level. Assignment decisions favor nodes with the highest reputation scores, subject to a diversity constraint that prevents all replicas from landing on nodes sharing a common failure domain. The Assignment Manager also monitors dataset health and initiates reassignment when a node's reputation falls below threshold or a node stops responding.

Storage nodes are the strategic players. Each node is a self-interested agent that provides storage capacity, runs the Prover for its assigned content, and chooses strategies to maximize expected net compensation.

Clients submit datasets at commitment time and receive identifiers for later retrieval. They interact with storage nodes only indirectly, through the assignment and health-reporting functions, and are not strategic players in the game analyzed here.

2.2 Strategy Spaces

Each storage node i chooses a strategy σ_i = (s_i, r_i), where $s_i \in [0,1]$ is the fraction of assigned blocks retained in local storage and r_i is a response policy mapping each incoming challenge to an action.

The soundness of the underlying audit protocol constrains r_i: a storage node cannot produce a valid proof for a block it does not hold. The response policy therefore reduces to a choice between honest proof generation (for locally held blocks), on-demand fetching (sourcing absent blocks from the network before the deadline), and abstention. The principal strategy classes are:

- – Honest ($s_i = 1$): retain all assigned blocks; respond promptly with valid proofs.
- – Partial retention ($s_i \in (0,1)$): retain a subset of assigned blocks; proxy or abstain for the remainder.
- – On-demand proxying ($s_i \approx 0$): retain no local copy; fetch all challenged blocks from the network before each deadline.
- – Exit: cease operation entirely.

We do not model cryptographic adversaries. The security of the underlying audit protocols against computationally bounded provers is established in the source literature and taken as given. Our concern is with rational deviators who select among the strategies above to maximize expected compensation net of storage cost.

2.3 Node Utility

Storage node i's utility over T audit rounds is:

$$U_i(\sigma_i) = \Sigma_{k=1}^{T} E[P_i(k)|\sigma_i] - C(s_i) \cdot T$$

where E[P_i(k) | σ_i] is the expected payment in round k under strategy σ_i (defined in §5) and C(s_i) is the per-round cost of maintaining local storage fraction s_i. C is increasing in s_i: storing more blocks locally costs more.

The operator's mechanism design objective is to choose P_i and the reputation update rules such that U_i is maximized at $s_i = 1$ for every storage node, regardless of the strategies chosen by other nodes; that is, honest storage is a dominant strategy.

2.4 Information Structure

The game has the following information structure:

- – Common knowledge: the payment function P_i, reputation update parameters (α, γ, λ), challenge parameters (C, Δt_max), de-listing threshold R_{min}, and diversity constraint D_{min}.
- – Publicly observable: each node's current reputation score R_i.
- – Private to each node: its true storage fraction s_i and its operational costs C(s_i).
- – Observed by the Verifier: proof validity $v_i \in$ {valid, invalid, absent} and response latency Δt_response,i for each audit round.

- – Not directly observed: the storage strategy s_i itself. The Verifier sees only the outcomes that s_i produces.

The central information asymmetry is that storage nodes have private information about their strategies while the Verifier observes only audit outcomes and response latency. The mechanism must translate these imperfect signals into economic consequences sufficient to make honest storage the dominant response to that asymmetry.

3 The Audit Loop

The operator runs a continuous audit loop for each active storage node. The loop has three phases: Challenge, Proof, and Verification. Together they constitute both the integrity audit and the liveness monitor for each node.

3.1 Challenge Phase

At each audit interval, the Challenger selects a random challenge for each active storage node. For nodes assigned content under a sparse-identifier protocol, the challenge is constructed by drawing a seed s uniformly at random and computing, for each TagBlock T across all assigned TagLists:

$$order(T, s) = H(id(T) \| s)$$

where id(T) is the block identifier of T. The C TagBlocks with the smallest order values form the challenge set. This construction is cross-dataset: it samples uniformly from the node's full assigned portfolio regardless of how many distinct stored datasets it spans. Only the seed s and the count C need to be transmitted; the Prover can reconstruct the full challenge set given its assigned TagLists.

The Challenger records the challenge issue time t_{issued} and the deadline $t_{deadline} = t_{issued} + \Delta$t_max, where Δt_max is a protocol parameter set to reflect the expected round-trip latency of an honest node with local storage. Parameter selection is discussed in §7.3.

3.2 Proof Phase

Upon receiving the challenge, the Prover reconstructs the challenge set, resolves each block identifier against its local content store, and applies the protocol's proof aggregation procedure. The resulting proof is returned to the Challenger. A Prover that cannot resolve a challenged block identifier, because the corresponding block was not retained locally, cannot produce a valid proof for that block. The underlying protocol's soundness guarantee ensures that a computationally bounded Prover cannot forge a valid proof for absent content.

3.3 Verification Phase

Upon receiving the Prover's response, or upon deadline expiry without a response, the Verifier records three values: the proof validity $v_i \in$ {valid, invalid, absent}, the response latency Δt_response,i = $t_{received}$ - t_{issued} (or ∞ for non-response), and the round index k. Proof validity and response latency are combined into a per-round score that feeds the node's reputation update (§4.1). Non-response is treated as a failed proof with infinite latency, the worst possible outcome on both dimensions.

3.4 Liveness Detection

A storage node that has gone offline will produce a sustained sequence of non-responses. The Verifier classifies a node as non-responsive after N_{fail} consecutive absent proofs. Upon this classification, the Assignment Manager immediately suspends payment and initiates reassignment: each dataset assigned to the node is migrated to a replacement node selected by the health-maximizing algorithm (§4.3). A node that comes back online may be re-admitted subject to a probationary challenge period under a clean reputation state.

4 Dual Scoring

4.1 Node Reputation Score

Each storage node i maintains a reputation score $R_i \in [0, 1]$. At each audit round k, R_i is updated based on the round outcome.

For a valid proof received within deadline:

$$R_i(k+1) = R_i(k) + \alpha \cdot (1 - R_i(k)) \cdot w(\Delta t_r esponse, i)$$

For an invalid proof or non-response:

$$R_i(k+1) = R_i(k) \cdot (1 - \gamma)$$

where $\alpha \in (0, 1)$ is a learning rate governing the speed of reputation recovery, $\gamma \in (0, 1)$ is a decay factor applied on failure, and w(Δt) is a latency weight:

$$w(\Delta t) = \exp(-\Delta t/\lambda)$$

A node that consistently produces valid proofs at low latency sees R_i converge toward 1. A single failure causes a multiplicative reduction proportional to γ. Latency-penalized valid proofs still increase reputation, but more slowly than low-latency proofs. Reputation is bounded between 0 and 1 by construction.

If R_i falls below a threshold R_{min}, the node is de-listed: it receives no new assignments and its existing assignments are migrated. De-listing is reversible; a node that completes a probationary re-entry sequence may be re-listed.

4.2 Dataset Health Score

Each stored dataset c is assigned to a set of k storage nodes $N_c = \{i_1, \ldots, i_k\}$. The health score of c is:

$$H_c = min_{i \in N_c} R_i$$

Using the minimum rather than the mean reflects the redundancy structure: a dataset is no more secure than its least-reliable assigned node. A single failing node degrades H_c regardless of the others' performance, correctly signaling that reassignment may be warranted. $H_c < H_{min}$ triggers reassignment of the lowest-scoring node in N_c.

4.3 The Assignment Manager

The Assignment Manager maintains the full mapping from datasets to storage nodes and continuously optimizes it toward maximum dataset health. Its decisions are governed by two objectives: placing newly committed content on nodes most likely to sustain it, and reassigning content away from degrading nodes before health falls below threshold.

For a new assignment, the Assignment Manager selects k nodes from the active pool to maximize the expected health score of the new assignment. The primary selection criterion is reputation: nodes are ranked by their current reputation score, reflecting accumulated audit performance. A secondary criterion is topological diversity: where information is available, assigned nodes should represent distinct failure domains to bound the probability of correlated failure. A single actor controlling multiple nodes in the same assignment set reduces effective redundancy; the diversification constraint limits this exposure.

For reassignment, the Assignment Manager selects a replacement node with $R_i > R_{min}$ and sufficient topological distance from the remaining assigned nodes, stores the content on the replacement, and removes the degraded node from the assignment set once the replacement has passed an initial challenge round.

The diversification constraint requires that assigned nodes not share known failure domains. Any available topology signals can inform this assessment, but such information may be incomplete or absent for many nodes.

In the absence of explicit topology information, the distribution of nodes around a ring-structured keyspace provides a useful approximation. Node identifiers occupy positions in a circular keyspace, and the participating population tends to be globally distributed across that space. Selecting assigned nodes from positions spread around the ring therefore tends to distribute replicas across diverse physical locations and administrative domains. An adversary seeking to control all k replicas of a dataset would need to hold k evenly-distributed positions in the ring, which requires controlling a proportional share of the total participating population.

The motivation for diversity is that assigned nodes sharing a failure domain reduce the effective redundancy of the assignment set. A single event affecting a shared failure domain, whether a network partition, a power outage, or a common operator choosing to defect, can produce correlated failures across multiple replicas simultaneously. From the verifier's perspective, this reduces the effective redundancy below k, undermining the statistical independence assumed by the detection probability analysis of §6. Distributing assignments across independent nodes is the proactive defense against this reduction; correlated failure detection (§6.5) is the reactive mechanism.

5 Incentive Mechanism

5.1 Payment Function

A payment function that produces the desired incentive structure must satisfy three properties. First, participation must be rational: a node operating honestly should earn more than the cost of storing its assigned content, so that joining the protocol as a persistent storage provider is a positive-value decision. Second, honest behavior must dominate the available alternatives: on-demand proxying, partial retention, and non-response should each yield strictly lower expected compensation than honest local storage. Third, reputation should translate directly into earnings: a node should be continuously rewarded for building and maintaining a high reputation, not merely penalized for falling below a minimum threshold.

These properties jointly determine the structure of the payment function. Compensation is paid for each audit round in which a node returns a valid proof:

$$P_i = B_{base} \cdot \exp(-\Delta t_{response,i}/\lambda) \cdot f(R_i)$$

where B_{base} is a base rate set to exceed the amortized per-round cost of honest storage; $\exp(-\Delta t_{response,i}/\lambda)$ is a latency decay factor that approaches 1 as response time approaches zero and falls toward $\exp(-\Delta t_{max}/\lambda)$ as response time approaches the deadline; and $f(R_i)$ is a reputation multiplier, $f : [0, 1] \to [f_{min}, 1]$, with $f(R_i) = f_{min} + (1 - f_{min}) \cdot R_i$.

Property A is satisfied when B_{base} exceeds the amortized storage cost per round: a node with even moderate reputation earns a positive margin on honest operation. Property B is satisfied through the latency decay factor: a node proxying data on demand incurs network round-trip latency on every challenge, which the exponential decay translates into substantially lower payment per round, and a node that fails or skips a round receives nothing and suffers a reputation penalty that reduces $f(R_i)$ in all subsequent rounds. Property C is satisfied by the reputation multiplier: since f is strictly increasing in R_i, every round in which a node performs well increases its earnings in all future rounds, creating a continuous incentive to maximize reputation rather than merely sustain it above a minimum threshold.

5.2 Payment Withholding

The baseline enforcement mechanism analyzed here is payment withholding: a node receives compensation only for rounds in which it returns a valid proof. Failed rounds yield no payment, and the associated reputation penalty reduces earnings in all subsequent rounds. The long-run value of participation therefore depends on sustained honest performance: a node that builds and maintains high reputation earns more per round and receives more assignments than one that does not. Whether payment withholding alone is sufficient to make honest storage the dominant strategy across all deployment contexts, or whether explicit collateral commitment is also warranted, is examined in §7.1.

6 Incentive Analysis

The mechanism design objective of §2.3 is to choose P_i and the reputation update rules such that honest storage, specifically $s_i = 1$ with prompt proof responses, maximizes U_i for every storage node regardless of other nodes' strategies. We show this holds against each strategy class in §2.2 by demonstrating that U_i(honest) > $U_i(\sigma$') for every deviation σ'.

6.1 On-Demand Proxying ($s_i \approx 0$)

A node playing the proxying strategy retains no local copy and fetches challenged blocks from the network before each deadline. Since the soundness of the audit protocol prevents forging proofs for absent blocks, proxying can produce valid proofs, but not at low latency. Network propagation adds a round-trip to every response, and under load, fetch latency may exceed Δt_max entirely.

The utility gap between honest and proxying strategies is:

$$U_i(honest) - U_i(proxying) = T \cdot B_base \cdot [\exp(-\Delta t_honest/\lambda) - \exp(-\Delta t_proxy/\lambda)] \cdot f(R_i) - [C(1) - C(0)] \cdot T$$

The first term is the payment advantage of honest storage, growing with the latency gap Δt_proxy - Δt_honest. The second term is the storage cost advantage of proxying. Because λ is calibrated so that fetch-consistent latency yields substantially reduced payment (§7.4), and because C(1) reflects commodity storage costs, the payment advantage dominates for any reasonable storage market. Proxying is a dominated strategy.

6.2 Partial Retention ($s_i = 1 - f,\ f > 0$)

A node retaining fraction (1-f) of its assigned blocks passes each audit round with probability (1 - P(f, N, C)), where P(f, N, C) is the hypergeometric detection probability. Each failed round yields zero payment and a multiplicative reputation penalty of (1-γ), compounding over time to degrade the reputation multiplier f(R_i).

The expected utility of partial retention over T rounds is:

$$U_i(partial) \approx T \cdot (1 - P(f, N, C)) \cdot B_base \cdot \exp(-\Delta t/\lambda) \cdot f(R_i(f)) - C(1 - f) \cdot T$$

where f(R_i(f)) is the reputation multiplier at the equilibrium reputation level reached under strategy f. Compared to honest storage:

$$U_i(honest) - U_i(partial) = T \cdot B_base \cdot \exp(-\Delta t/\lambda) \cdot [f(R_i(1)) - (1 - P(f, N, C)) \cdot f(R_i(f))] - [C(1) - C(1-f)] \cdot T$$

The first term is positive and growing: each failure degrades R_i geometrically at rate (1-γ), widening the reputation multiplier gap over time. For partial retention to be profitable this difference must be negative: the storage savings must exceed the compounding payment and reputation losses. Calibrating B_{base} and γ so this condition fails for all economically meaningful f ensures partial retention is dominated. Parameter selection is discussed in §7.4.

6.3 Exit and Non-Response

A node that exits the protocol after round k forgoes all future payments and storage costs from round $k+1$ onward. The utility gap relative to continued honest operation is:

$$U_i(\text{honest}) - U_i(\text{exit at } k) = \sum_{t=k+1}^{T} \left(E[P_i(t) \mid \text{honest}] - C(1)\right)$$

Each term is positive when honest operation yields positive per-round profit, that is, when $B_{\text{base}} \cdot \exp(-\Delta t/\lambda) \cdot f(R_i) > C(1)$. For any node whose expected per-round earnings exceed its per-round storage cost, exit before T is strictly dominated by continued honest operation.

When this condition does not hold, because reputation has degraded sufficiently that $f(R_i)$ depresses earnings below the cost floor, or because the node holds too few assignments to amortize fixed costs, exit is rational. The mechanism accommodates this outcome by design: a de-listed or exiting node has its assignments migrated to higher-performing nodes, and the assignment pool self-selects toward participants for whom honest storage remains economically viable. The enforcement objective, that honest storage dominates all deviation strategies, applies within the participation regime where individual rationality holds; sustaining that regime broadly requires calibrating B_{base} to provide a meaningful margin above storage costs (§7.4).

Non-response is dominated by clean exit within this regime. A node that goes silent rather than exiting explicitly receives zero payment for each non-responsive round and accrues a multiplicative reputation penalty $(1-\gamma)$ per absent round. After N_{fail} consecutive non-responses the node is de-listed and its assignments are migrated, forfeiting the same future payment stream as clean exit while additionally incurring reputation damage that raises the cost of re-entry. A node for whom continued participation is individually rational therefore strictly prefers clean exit over silence, and strictly prefers honest operation over clean exit.

6.4 Sybil Resistance

A single operator controlling k storage node identities does not improve its utility by multiplying identities. Each identity is scored and paid independently under U_i, so the operator's aggregate utility is $\Sigma j\ U\{i_j\}$ across its k identities. If all identities store honestly, the aggregate equals what a single honest node would earn for the same content. If some identities reduce local storage, those identities individually face the utility loss of §6.2; the sybil multiplication neither reduces per-identity detection probability nor shifts the dominant strategy from honest storage.

6.5 Collusion

A coalition of nodes that partitions an assigned dataset, each holding only a fraction, places each coalition member in the partial-retention scenario of §6.2. Node i holding fraction (1-f_i) of its assigned share fails at rate P(f_i, N, C) and accrues the same payment and reputation losses as a solitary partial-retention deviator. Collusion does not change any individual node's dominant strategy: honest storage of each node's full assigned share remains the utility- maximizing choice per identity.

Beyond the per-node analysis, correlated failures within an assignment set provide a joint signal to the Assignment Manager. Nodes that share a hidden failure domain or coordinate omissions exhibit temporally correlated audit failures, identifiable through pairwise failure- correlation tracking (§4.3). Sustained correlation triggers reassignment regardless of individual reputation scores, raising the cost of sustaining a coalition above the per-node penalties alone.

7 Discussion and Open Questions

7.1 Collateral Commitment vs. Payment Withholding

Payment withholding ties the cost of defection to the value of foregone future payments. Its effectiveness depends on two conditions: that honest nodes have a sufficiently long time horizon (they value the future payment stream, not only immediate payouts), and that B_{base} is set high enough relative to storage costs that the future payment stream is worth protecting. Under these conditions, the threat of losing ongoing income, analogous to an implicit reputation stake, may be sufficient to deter defection without requiring explicit collateral.

Collateral commitment supplements payment withholding by requiring nodes to post a deposit in advance, a portion of which is forfeited upon proven non-compliance. This provides a deterrent that does not depend on the node's time horizon and is robust against nodes planning to exit. The collateral may be held by the operator, by a mutually agreed escrow agent, or under any other arrangement the parties accept; the key requirement is that the conditions for forfeiture are defined clearly in advance and that the node cannot unilaterally withdraw the deposit during an active assignment. The trade-off relative to payment withholding is operational complexity: a commitment mechanism must be established, the forfeiture conditions must be specified, and the deposit amount must be calibrated to exceed the expected gain from defection.

The appropriate enforcement mechanism likely depends on deployment context. For a closed network with vetted storage nodes and ongoing contractual relationships, payment withholding may be sufficient. For an open network where nodes can join and exit freely, collateral commitment provides a deterrent that payment withholding alone cannot.

7.2 The Diversity Verification Problem

The failure domain hierarchy of §4.3 assigns a diversity distance to each pair of storage nodes, but most of the hierarchy's upper levels, specifically organizational separation and jurisdictional independence, are not directly observable. The system can verify network-layer diversity (ASN, IP prefix, latency-inferred location) through external data sources that are difficult to systematically falsify. Organizational independence cannot be verified externally at all.

Cloud provider concentration is the primary gap. A single cloud account can deploy storage nodes across many autonomous systems and geographic regions, producing node pairs with $D(i,j) \geq 4$ by every observable metric while remaining under unified organizational control. An account-level event (a provider banning the account, a credential compromise, or a legal order) can disable all such nodes simultaneously regardless of their apparent network diversity.

The practical consequence is that the declared k-of-n redundancy of an assignment set may exceed

the effective redundancy by an unknown factor. The system cannot determine the true effective k from observable data alone, and therefore cannot guarantee the stated health threshold H_{min} with certainty, only under the assumption that declared diversity reflects true independence.

Correlated failure patterns are the post-hoc detection mechanism. Nodes sharing a hidden failure domain will exhibit temporally correlated audit failures even when declared independent. The Assignment Manager should maintain pairwise failure-correlation estimates across all nodes assigned to the same dataset and treat sustained correlation above a threshold as evidence of shared infrastructure, triggering reassignment regardless of declared diversity distance. This connects to the collusion detection argument of §6.5: the same correlation signal that suggests coordination among independent actors also suggests shared infrastructure among nominally distinct ones.

A collateral commitment layer (§7.1) provides a partial remedy at the organizational level. A collateral deposit functions as an observable identity anchor: it is costly to multiply because each identity requires committed capital. An operator wishing to control multiple nodes in the same assignment set must commit capital proportional to the number of identities, raising the economic barrier to undeclared concentration even when network-level independence cannot be verified.

7.3 Bounty-Based Peer Verification

Several audit protocol variants support public verification: any party holding only the public key can issue a valid challenge and verify the response, without access to private key material. This property enables an alternative economic model in which storage nodes audit one another rather than relying solely on operator-issued challenges.

Under a bounty model, any node that observes a valid challenge for which another node fails to produce a valid proof may submit the challenge-and-failure record to the operator and claim a bounty payment. This model distributes auditing workload and may increase audit frequency without proportional operator cost.

The model introduces a significant attack surface. An operator controlling two node identities, one functioning as reporter and one deliberately non-compliant, can farm bounty payments by staging failures. Whether the bounty exceeds the penalties absorbed by the non-compliant identity depends on calibration; if bounties are funded from a shared pool or independently of the non-compliant node's forfeit, the attack can be profitable at scale.

Mitigations include requiring reporters to hold active assignments with their own collateral at risk, rate-limiting bounties per reporter per time window, and requiring cryptographic proof that the reporter independently holds assigned content rather than having merely forwarded a challenge. None of these fully eliminates the attack surface. The bounty model warrants further mechanism design work before deployment.

7.4 Parameter Selection

The mechanism involves several tunable parameters with direct consequences for deterrence effectiveness and node economics.

The challenge size C governs the per-round detection probability P(f, N, C). Larger C increases detection sensitivity at the cost of greater Prover computation per round. For a corpus of N blocks

and a target detection probability P* at corruption fraction f, *C should satisfy P(f*, N, C) $\geq$ P*; the hypergeometric formula gives the required C directly.

The latency parameters Δt_max and λ must be calibrated to the expected distribution of honest response latencies. Setting Δt_max too tightly penalizes honest nodes on high-latency links; setting it too loosely allows proxying nodes to satisfy the deadline. λ should be set such that responses consistent with local storage yield exp(-Δt / λ) $\approx$ 1, while responses consistent with on-demand fetching yield substantially reduced payment.

The failure penalty γ and recovery rate α together determine how quickly reputation degrades after a failure and how quickly it recovers. γ should be large enough that sustained partial retention drives R_i toward R_{min} within a manageable number of rounds, while α should allow a node experiencing a transient failure (network outage, hardware fault) to recover without permanent penalty.

The base payment B_{base} must exceed the amortized cost of storing one unit of content per audit interval, providing a positive margin to honest nodes. That margin must also exceed the expected payment saved by any deviation strategy at the equilibrium failure rate implied by that strategy.

7.5 Multi-Provider Extensions

A client requiring k-of-n availability, meaning the ability to reconstruct the dataset from any k of n storage nodes, can combine the audit framework with an erasure-coding scheme. The dataset is encoded into n shares, each assigned to a distinct storage node; any k shares suffice for reconstruction. The TagList model extends naturally: each node receives a TagList for its assigned share rather than the full dataset.

Dataset health in this setting reflects not the minimum reputation among assigned nodes but the probability that at least k of n nodes remain compliant, a threshold reliability function over the n reputation scores. The Assignment Manager's objective becomes maximizing this threshold reliability. The redundancy structure also changes the partial-retention economics: a node that discards its share is not detectable through cross-node correlation unless failing nodes exceed n-k simultaneously.

8 Conclusion

Cryptographic audit protocols establish that a storage node possesses its assigned content at audit time; they do not compel possession. Translating a verification capability into an enforcement mechanism requires an economic layer that makes honest storage more profitable than any deviation available to a rational node.

The framework proposed here closes this gap through three coupled mechanisms: a payment function that rewards low-latency proof responses and scales with reputation, making on-demand proxying and partial retention economically inferior to honest local storage; a dual scoring model that tracks both node-level reliability and dataset-level health, enabling the Assignment Manager to continuously route content toward nodes most likely to sustain it; and an audit loop that doubles as a liveness monitor, triggering automatic reassignment when nodes go silent.

The central result is conditional: provided parameters are calibrated appropriately and the node

values continued assignment membership, honest, low-latency local storage is the dominant strategy for a rational storage node. The result holds not because defection is cryptographically impossible, but because the payment structure and reputation dynamics make defection unprofitable. The probability of detection per round, compounded over time and combined with payment withholding and reputation decay, causes any deviation strategy to yield lower expected compensation than honest behavior under these conditions. When the individual-rationality condition does not hold, because reputation has degraded or assignment volume is insufficient to cover costs, exit is rational and the mechanism accommodates it through reassignment.

Open questions remain around the enforcement mechanism (whether payment withholding alone is sufficient or whether collateral commitment is warranted), the sybil attack surface introduced by bounty-based peer verification, and the extension of the framework to multi-provider erasure-coded deployments. These are identified as the principal directions for future work.