Proof of Data Possession for IPFS: Verifying Remote Storage Reliability in a Decentralized Network

2026

1 Abstract

The InterPlanetary File System (IPFS) functions primarily as a decentralized content distribution network, leveraging content-derived identifiers (CIDs) to guarantee data integrity. Users benefit from a powerful mechanism: the immutability and verifiability of the Merkle Directed Acyclic Graph (DAG). When content is retrieved from any node, the client can cryptographically validate the data against its CID, confirming its structural integrity irrespective of the source's trustworthiness. This inherent feature of IPFS is highly valuable.

However, IPFS is fundamentally a content distribution layer, not a guaranteed persistence layer. For data to remain continuously available, content must be actively retained on a node that stays online; in IPFS terminology, this is referred to as pinning. Pinning may be performed by the content owner on their own infrastructure, or delegated to a third-party storage provider. A class of such providers expose a standardized HTTP API, defined by the IPFS Pinning Service API specification, and are commonly referred to as pinning services. Whether self-hosted or third-party, the persistence of pinned content ultimately depends on the continued operational commitment of whoever operates the pinning node.

The critical challenge arises when scaling data validation to massive datasets. While IPFS guarantees integrity that the data, if retrieved, is untampered, it does not guarantee availability. To confirm that a pinning service has faithfully retained a large dataset, the current practical validation mechanism, re-downloading the entire dataset to re-calculate the Merkle DAG, is prohibitively expensive and computationally infeasible for archives measured in terabytes or petabytes. Consequently, the verification of persistence for large-scale decentralized storage remains an assurance problem anchored in trust.

To close this gap, we establish that CID-addressed Merkle DAG blocks are a valid instantiation of the sparse-identifier block-store model required by PDP/POR protocols that support non-sequential block addressing, and we provide a general framework through which such protocols can be applied to IPFS with no modification to their cryptographic cores. IPFS identifies blocks by content-derived hashes of arbitrary byte length, while PDP/POR protocols traditionally operate on integer-indexed arrays. We prove the instantiation sound for sparse-capable protocols (Theorem 1, §6.2), so the security reductions of the underlying protocols carry through unchanged. Protocols requiring sequential integer positions (Erway, BJO) are also supported by the framework but operate on a

per-dataset basis rather than across the full pinned inventory. The framework is structured around four functional roles: a Tagger that generates cryptographic verification metadata at commitment time, a Challenger that issues periodic audit challenges, a Prover that responds with compact proofs, and a Verifier that validates responses and updates storage reliability assessments.

The framework enables compact remote audits of large pinned datasets. The communication cost of each audit round is fixed regardless of dataset size, comprising a challenge of approximately 41 bytes and a proof ranging from 80 to 288 bytes. Furthermore, for protocols supporting non-sequential block identifiers, a single challenge round can sample uniformly across all content pinned to a storage node, auditing the node's entire storage commitment in one round-trip.

We define the foundational `TagBlock` and `TagList` data structures that realize this mapping, detail the setup procedure required to traverse the Merkle DAG and generate them, and demonstrate the framework against several protocol variants from the literature. The security of each construction reduces to the security of the underlying cryptographic protocol, introducing no new cryptographic assumptions.

The reliability provided by this verification scheme is probabilistic: a single round detects the loss of a fraction f of N blocks with a hypergeometric probability $P(f, N, C)$, where C is the challenge size. Critically, the cumulative detection probability across k sequential rounds approaches 1.0 monotonically.

In summary, our technology not only provides a mathematically rigorous method for validating massive datasets without requiring full download but also opens the possibility for deploying a distributed pinning service layer that significantly reduces reliance on single, trusted entities. The economic enforcement mechanisms required to integrate this protocol into a fully trustless, incentivized storage network are identified for future work.

2 1. Introduction

The InterPlanetary File System (IPFS) is fundamentally a content-addressed distribution protocol. Content is added to a local node and assigned a root Content Identifier (CID), which is cryptographically derived from the data's hash. This CID provides an inherent integrity guarantee: any retrieved block that does not match its recorded hash is immediately detectable, ensuring data immutability upon retrieval. However, IPFS, in its base form, does not guarantee data permanence. Once the publisher's node goes offline, the data becomes unavailable to the network unless another peer has independently retained it. While this model is sufficient for ephemeral content broadcasting, it is insufficient for long-term archival storage.

Pinning services solve this problem by acting as specialized, persistent custodians. A pinning service accepts a root CID and commits to maintaining a continuous replica of the corresponding Merkle DAG. By continuously advertising the data in the Distributed Hash Table (DHT), they extend content availability and transform the system from a pure distribution protocol into a functional storage layer. In this sense, pinning services serve as the necessary storage backbone for IPFS.

The inherent vulnerability lies in the fact that the entire operational model relies on a relationship of trust. IPFS's cryptographic content addressing only verifies the integrity of data received from the network. Crucially, it provides no protocol-level mechanism to verify that a remote node is

actively retaining a complete dataset. For a client managing a multi-terabyte archival dataset, downloading the entire corpus, recomputing the Merkle root and confirming integrity, is prohibitively expensive in terms of bandwidth, time, and computational resources. Consequently, in practice, clients lack an efficient, protocol-level mechanism to confirm the faithful retention of a large dataset by a remote provider short of a full retrieval. Assurance is currently limited to external measures such as contractual agreements or provider reputation.

This paper addresses this systemic gap by adapting protocols from two families established in the cryptographic literature: Provable Data Possession (PDP) and Proofs of Retrievability (POR). These are families of protocols, not specific constructions; each family admits multiple instantiations with different trade-offs. In all cases, the verifier obtains compact, probabilistic evidence that a remote server holds a complete dataset by exchanging a challenge and a proof, without retrieving the data. The verifier's computational complexity is sublinear in the dataset size. Applying these protocols to IPFS requires resolving a structural mismatch: IPFS organizes data as a Merkle DAG of content-addressed leaf blocks (identified by CIDs), while PDP and POR protocols traditionally operate on arrays indexed by sequential integers. We demonstrate that this structural bridge is feasible for the subset of protocols designed to support non-sequential block identifiers, and we formally define the requisite data structures, `TagBlock` and `TagList`, that realize this translation.

Our primary contributions are:

1. **Formal model and main theorem.** We formalize CID-addressed IPFS Merkle DAG blocks as a sparse-identifier block store and prove (Theorem 1, §6.2) that this instantiation satisfies the security requirements of every sparse-capable PDP/POR protocol. The security reductions of the underlying protocols carry through unchanged.
2. **TagBlock and TagList data structures.** We define the data structures that realize the mapping from IPFS content-addressed blocks to the block-store interface PDP/POR protocols require, and describe the DAG traversal procedure that generates them at setup.
3. **Cross-pin challenge construction.** We propose a challenge-response protocol that operates across the union of all content pinned to a storage node, auditing the node's entire storage commitment in a single round-trip. This construction depends on sparse-identifier support and is specific to the IPFS pinning context.
4. **Framework validation across protocol families.** We instantiate the framework with five protocol families covering the key design axes: static vs. dynamic data (Ateniese S-PDP, Erway D-PDP I), private vs. public verification (SW-Priv, SW-Pub, Erway), and possession vs. retrievability guarantees (SW-Priv, SW-Pub, BJO). Each instantiation inherits its security from the underlying protocol; no new cryptographic assumptions are introduced.

This paper establishes cryptographic guarantees under the assumption that the verifier issues genuine independent challenges. Adversarial behaviors requiring coordination across multiple storage nodes, including collaborative fragmentation and Sybil-mediated sharding (§4.3, §4.7), are not addressed by the cryptographic layer.

3 2. Background

3.1 2.1 IPFS: Distribution vs. Storage

The core protocols of IPFS are distribution protocols. The Distributed Hash Table (DHT) allows peers to announce and discover content. Bitswap governs the exchange of blocks between peers on demand. HTTP gateways expose IPFS content to ordinary web clients. Together these mechanisms make content available across the network while it is being actively served, but they carry no obligation of persistence. A block that no peer chooses to retain will eventually become unreachable.

Pinning introduces a distinct operational layer. When a node pins a dataset, it commits to retaining and continuously advertising it regardless of demand. This transforms the node from a transient participant in content distribution into a durable storage provider. Pinning services, which expose a standardized API for this purpose, effectively supply the storage layer that the base IPFS protocol does not: they keep content available after the original publisher has gone offline and independent of ongoing retrieval activity.

Distribution and storage are therefore separate concerns in the IPFS ecosystem, served by separate mechanisms. The protocols surveyed in this paper operate at the storage layer: their purpose is to verify that a pinning node is faithfully retaining assigned content, a question that the distribution protocols have no means to answer.

3.2 2.2 Pinning Services and the Trust Gap

A pinning service accepts a root CID from a client and commits to maintaining a replica of the corresponding DAG, keeping it available to the IPFS network for an agreed period. Pinning services expose a standardized interface (the IPFS Pinning Service API) compatible with major IPFS implementations. By maintaining persistent replicas and DHT advertisements, they supply the permanence layer that the base IPFS protocol does not provide: content remains retrievable after the original publisher goes offline.

Consider a research group that has assembled a large machine learning training corpus and published it to IPFS. The root CID is shared with the community; reproducibility of published results depends on the dataset remaining available and intact. The group pins the dataset with a provider and resumes other work. Nothing in the IPFS protocol will alert the group if the provider silently loses, corrupts, or deletes blocks during a period of inactivity. Confirming completeness requires downloading and reconstructing the entire corpus, fetching every leaf block and recomputing the Merkle root from the bottom up, to verify that the result matches the known root CID. For a dataset of substantial size, the communication and computational cost of this operation scales linearly with the dataset, making continuous or frequent integrity checks prohibitively expensive relative to the cost of ordinary use.

This is the trust gap at the core of the pinning service model. IPFS content addressing verifies that any block actually retrieved from the network matches its CID. It does not verify that a block still exists on a remote node. The two guarantees are categorically different: one confirms what you have received, the other confirms what the server currently holds. Cryptographic integrity upon retrieval is not the same as cryptographic proof of remote possession. Clients extend trust implicitly, treating the provider's continued operation and public reputation as a proxy for continued faithful

storage. There is no protocol-level mechanism to distinguish a provider honestly retaining every block from one that has quietly deleted a fraction of the dataset and is satisfying retrieval requests by fetching missing blocks on demand from the wider network.

3.3 2.3 Provable Data Possession and Proofs of Retrievability

Provable Data Possession (PDP) and Proofs of Retrievability (POR) are cryptographic protocols designed to close this gap: convincing a verifier that a remote server holds a complete copy of outsourced data, without requiring the verifier to retrieve it. Both families share a common structure. During a setup phase, the client (or a trusted coordinator) preprocesses the data by computing cryptographic tags, which are compact metadata bound cryptographically to each block. During an audit phase, the verifier issues a challenge specifying a random subset of block indices; the server responds with a compact proof computed from those blocks and their tags. The verifier checks the proof in time sublinear in the size of the dataset.

The two families differ in what they guarantee and who can verify. PDP schemes demonstrate that the server possesses the challenged blocks at audit time; they make no assertion about whether data lost beyond the sampled blocks is recoverable. POR schemes additionally incorporate error-correcting codes or homomorphic authenticators to ensure retrievability, the property that sufficiently many successful audit responses imply the complete dataset can be reconstructed. Along a separate axis, private verification schemes require the original client to hold secret material, restricting auditing to the data owner. Public verification schemes allow any party holding a public key to issue challenges and verify responses, enabling delegated or third-party auditing without exposing the underlying data.

Section 3 surveys the specific protocol constructions this work builds upon, comparing their properties along these dimensions and others relevant to the IPFS setting.

4 3. Protocol Families

This paper does not propose new cryptographic constructions. Instead, it identifies five representative members of the PDP and POR literature that collectively cover the design space relevant to IPFS pinning, and uses them as instantiation targets for the framework developed in §5. Section 6 proves that each protocol preserves its original security guarantees when operating over CID-addressed storage. The selection spans the key dimensions of the protocol space: static vs. dynamic data, private vs. public verification, and possession-only vs. retrievability guarantees. Table 1 summarizes the capabilities relevant to the IPFS setting.

Table 1: Protocol capabilities. "Sparse IDs" indicates whether the protocol accepts arbitrary byte-string block identifiers rather than requiring sequential integer positions. *Erway verification requires only the public key and basis; if the basis is published, any party can verify.

Protocol	Verification	Sparse IDs	Extraction	Proof size
Ateniese S-PDP	Private	Yes	No	288 B (constant)
Erway D-PDP I	Public*	No	No	$O(C \log N)$
SW-Priv	Private	Yes	Yes	80 B (constant)

Protocol	Verification	Sparse IDs	Extraction	Proof size
BJO	Private	No	Yes	82 B (constant)
SW-Pub	Public	Yes	Yes	192 B (constant)

4.1 3.1 Ateniese S-PDP

Ateniese et al. introduced Scalable Provable Data Possession (S-PDP) in "Provable Data Possession at Untrusted Stores" (ACM CCS 2007). S-PDP is a static PDP scheme: it proves possession of a file that does not change after setup. The client computes an RSA-based cryptographic tag for each block and uploads the blocks and tags to the server, retaining only a key pair. During an audit, the verifier issues a challenge over a random subset of blocks; the server aggregates the corresponding tags and block values into a single compact proof using the homomorphic structure of the RSA group. The aggregated proof is constant in size regardless of how many blocks are challenged (288 bytes in our implementation). Verification requires the client's secret blinding factor, making it private: only a party holding the secret can audit.

Setup cost is dominated by RSA key generation and one modular exponentiation per block, making it the most expensive scheme at setup among those studied. Proof generation cost grows linearly with challenge size at a shallow slope, and proof wire size is constant.

4.2 3.2 Erway D-PDP I

Erway, Küpçü, Papamanthou, and Tamassia extended S-PDP to support provable dynamic updates (insertions, modifications, and deletions) in "Dynamic Provable Data Possession" (ACM CCS 2009). D-PDP I replaces the static tag structure with a rank-based authenticated skip list. The client retains only a 32-byte basis, the hash label of the skip list's root, as its entire local state; all per-block metadata is held by the server. Update operations run in expected $O(\log n)$ time and produce a basis update the client can verify from $O(\log n)$ server-provided data.

D-PDP I requires no secret verification key: block tags are publicly computable, and both verification steps, reconstructing the skip-list root hash and checking the aggregated proof against the public key, require only the public key and the basis. The basis is a collision-resistant hash commitment with no embedded secrets; if published, it enables any third party to verify audit proofs without access to private material. This contrasts with Ateniese, whose verification requires a secret blinding factor known only to the original client.

One consequence of the skip-list structure is that blocks must be addressed by sequential integer rank. D-PDP I does not support arbitrary byte-string block identifiers. Proof size also grows $O(\log n)$ per challenged block, making Erway the only scheme studied whose proof wire size scales with challenge size.

4.3 3.3 Shacham-Waters Private POR (SW-Priv)

Shacham and Waters introduced two POR constructions in "Compact Proofs of Retrievability" (ASIACRYPT 2008). The private-key variant (§3.2 of the paper) uses PRF-keyed tags over a large prime field Z_P, with each block decomposed into a fixed number of sectors. The client retains a

compact secret key; verification requires that key, making the scheme strictly private. Each audit response consists of a constant number of field elements (one aggregated tag value and one linear combination per sector), regardless of file size (80 bytes in our implementation). Proof generation and verification are both effectively instantaneous.

SW-Priv supports an unlimited number of audits and accepts arbitrary integers as block identifiers, enabling non-sequential addressing. It provides an extractor: the original file can be recovered by submitting a sufficient number of challenges and solving the resulting linear systems. Setup cost is far lower than the RSA-based schemes (microseconds per block versus milliseconds), because it requires only field arithmetic.

4.4 3.4 Bowers-Juels-Oprea POR (BJO)

Bowers, Juels, and Oprea improved on the original Juels-Kaliski construction in "Proofs of Retrievability: Theory and Implementation" (RSA Security 2009). BJO uses two layers of encoding: an outer systematic adversarial error-correcting code (SA-ECC) applied to the file, and an inner linear code over Z_P for challenge responses. At setup the client precomputes Q encrypted sentinel values and uploads them with the encoded file, consuming one per audit; the total number of audits is bounded by Q. Verification decrypts the sentinel returned in the proof using a secret key, making BJO strictly private.

BJO provides extraction: the full file can be recovered from a sufficient collection of audit responses by solving linear systems and decoding the outer ECC. The ECC encoding requires sequential block positions, so BJO does not support arbitrary block identifiers. Detection probability at small corruption fractions is modestly lower than the other schemes studied, a consequence of the inner code structure.

4.5 3.5 Shacham-Waters Public POR (SW-Pub)

The public-key variant of the Shacham-Waters construction (§3.3) replaces field arithmetic with pairing-based cryptography over BN-256. Block tags are BLS-style signatures in G_1:

$$\sigma_i = \alpha \cdot (H(\lambda \| id_i) + \textstyle\sum_j f_{i,j} \cdot u_j)$$

where α is a secret scalar whose corresponding public value $v = \alpha \cdot G_2$ is published. Verification evaluates two Ate pairings and requires no secret material; it is the only fully public scheme among those studied. Because $H(\lambda \| id_i)$ accepts an arbitrary byte string as id_i, the scheme supports non-sequential block addressing.

SW-Pub provides extraction on the same basis as SW-Priv and supports unlimited audits. Setup cost is moderate, substantially cheaper than the RSA-based schemes but more expensive than SW-Priv. Verification incurs pairing overhead and takes roughly an order of magnitude longer than SW-Priv, though proof wire size remains constant at 192 bytes.

5 4. Threat Model: Adversarial Strategies in Incentivized Peer-to-Peer Storage

The trust gap described in §2.2 has a precise economic structure. In an incentivized storage network, a rational node is rewarded for appearing to store data and penalized only when detected not storing it. If detection relies solely on retrieval requests or naive challenge-response protocols, a node can reduce its actual storage burden substantially while maintaining the appearance of compliance. IPFS's architecture, which is optimized for rapid content discovery and transient block transfer, makes several such strategies straightforward to execute: a node that holds no local copy of a dataset can still satisfy most retrieval requests by fetching blocks on demand from the wider network.

The following taxonomy describes the principal evasion strategies available to a rational adversary in this setting. Each represents a distinct mechanism by which a node can decouple apparent availability from durable local custody. Section 6 discusses how the protocol's challenge construction and latency-weighted scoring are designed to raise the cost of each strategy.

5.1 4.1 On-Demand Proxying

The node holds no local copy of the assigned dataset. When a retrieval request or audit challenge arrives, the node fetches the required blocks from other peers in real time and returns them as if stored locally. Because IPFS is designed for fast content routing, this strategy can satisfy many protocols within their latency budgets without the node ever committing to persistent storage.

5.2 4.2 Subset Retention (Hot-Block Caching)

Rather than storing the complete dataset, the node retains only a subset of blocks, typically those most frequently requested, and sources the remainder on demand. Over time this strategy converges toward a minimal local footprint that maximizes observed availability while substantially reducing storage overhead. The node appears fully compliant to any verifier that samples from the retained subset.

5.3 4.3 Collaborative Fragmentation

A coalition of distinct operators partitions a dataset among themselves, each holding only a fragment. When one node receives a retrieval or audit request for content it does not hold, it coordinates with coalition partners to reconstruct the required payload before responding. Each node in the coalition presents the appearance of full availability while none holds the complete dataset locally.

5.4 4.4 Transparent Relaying

The node operates purely as a proxy, forwarding all requests to a downstream storage provider and relaying responses back to the requester. The node contributes no local storage capacity. In practice, the downstream provider may be a commercial HTTP gateway, a different IPFS node, or an alternative storage network, any of which can serve blocks faster than typical audit timeout windows.

5.5 4.5 Ephemeral Caching

The node stores blocks in volatile memory or an eviction-bound cache upon first retrieval, holding them only long enough to serve subsequent nearby requests. Once cache pressure or an idle timer expires, the data is discarded. The node appears to possess data during any window that follows a recent retrieval, but provides no durable retention guarantee between windows.

5.6 4.6 Latency-Based Bypass

Where the audit protocol permits a generous response deadline, the node defers fetching any absent blocks until shortly before that deadline, sourcing them from the network just in time to produce a valid response. This strategy exploits the gap between network propagation latency (typically seconds) and audit timeout windows (which may be minutes or longer), substituting on-demand availability for local persistence.

5.7 4.7 Sybil-Mediated Sharding

A single operator controls multiple synthetic node identities and partitions the dataset across them, minimizing the storage commitment of each individual identity. Unlike collaborative fragmentation (§4.3), which involves coordination among independent parties, sybil-mediated sharding is operated by a single economic actor who captures the full storage payment while distributing the actual storage cost across fabricated identities. Externally, the identity set presents as a resilient multi-node infrastructure.

6 5. Protocol Design

This section instantiates the framework for IPFS. Sections 5.1 and 5.2 establish the formal correspondence whose soundness Theorem 1 (§6.2) formalizes, translating CID-addressed Merkle DAG blocks into the sparse-identifier block-store model that the protocol families of §3 require. Sections 5.3 through 5.6 define the TagBlock and TagList data structures and the setup, challenge, and proof procedures that make the mapping operational.

6.1 5.1 IPFS Blocks as a Sparse Array

In standard PDP and POR formulations, a file is divided into n blocks indexed by sequential integers $0, 1, \ldots, n-1$. A challenge names a subset of these positions; the server proves possession by seeking to each named position within the file and computing the protocol response from the bytes found there. The block identity is bound to a specific file at a specific offset. This model fits ordinary file storage naturally but poses a problem for IPFS: IPFS has no concept of a file offset. Every node in a Merkle DAG, whether a leaf block of raw bytes or an internal node encoding link structure, is identified by its CID, a cryptographic hash of its content. There is no canonical integer index.

The resolution is to present CID bytes directly as block identifiers to the underlying protocols. Each protocol is designed to accept arbitrary byte arrays as block identifiers and decode them as needed for cryptographic computation. Content whose data is arranged as a sequential array of blocks indexed by integers are represented in our system as an array of blocks whose IDs contain the binary encoding of the index. Content such as IPFS blocks who lack a natural integer index will have

a non-contiguous, non-sequential identifiers. The block-store abstraction each protocol operates against accepts arbitrary byte-string identifiers; CID bytes satisfy this interface without modification. More significantly, CID-based identification alters the semantics of block retrieval. Rather than seeking to a fixed offset within a contiguous file, the storage node resolves each identifier against its local content-addressed store, which maps each CID to the corresponding block bytes. The resolved content is determined entirely by the identifier itself, independently of which pinned dataset originally introduced the block, which ordinal position the block occupies within any particular collection, and whether the block is a leaf or an internal DAG node. Block identity is globally content-derived rather than locally positional.

This property enables what we term a cross-pin challenge: a single audit round whose block sample is drawn not from one pinned dataset but from the union of all datasets pinned to a storage node. In a standard PDP/POR deployment, the server proves possession of one file at a time, and a separate challenge must be issued per dataset to audit the full storage commitment. In our scheme, a coordinator that has pinned multiple datasets to the same storage node retains verification metadata for all of them. Because each block record carries only a content identifier and a cryptographic tag, with no file reference or positional offset, the coordinator can construct a single challenge drawing block records from any combination of pinned datasets. The server resolves each identifier independently against its local store and computes the proof response exactly as it would for a single-dataset challenge. One compact challenge round therefore audits the server's entire assigned storage portfolio simultaneously.

This capability is available for protocols that accept arbitrary byte-string block identifiers: Ateniese, SW-Priv, and SW-Pub. The remaining two protocols, Erway and BJO, require sequential integer positions: Erway because skip-list path proofs are position-dependent, and BJO because the ECC encoding is positionally structured. These protocols operate on a single TagList at a time and do not support cross-pin challenges.

6.2 5.2 Protocol Selection

The choice of protocol determines what the system can guarantee and who can verify it. Three dimensions matter for the IPFS setting.

Cross-DAG challenges. As described above, only protocols that accept arbitrary block identifiers (Ateniese, SW-Priv, and SW-Pub) can combine blocks from different pinned DAGs into a single challenge. Erway and BJO require a single sequentially-ordered block store.

Public verification. SW-Pub requires no secret material for verification: any party holding the public key can issue challenges and check proofs. This enables delegated or third-party auditing independent of the original client. Erway can also support public verification if the client publishes the basis (§3.2). The remaining protocols, Ateniese, SW-Priv, and BJO, are private: verification requires secret material held by the original client.

Extraction. SW-Priv, BJO, and SW-Pub provide extractability: sufficient audit responses allow the full dataset to be recovered without direct retrieval. This is relevant to data recovery scenarios where the storage provider's cooperation cannot be assumed. Ateniese and Erway offer possession guarantees only.

6.3 5.3 TagBlock and TagList

The TagBlock is the atomic metadata unit:

```
// TagBlock pairs a storage-proof tag with the IPFS CID of the block it authenticates.
type TagBlock struct {
    Tag line.Tag // opaque auth tag produced by the protocol
    Cid cid.Cid  // CID of the IPFS block whose bytes were tagged
}
```

The Tag field is an opaque byte slice whose format depends on the protocol: an RSA group element for Ateniese, a field element for SW-Priv, a G_1 point for SW-Pub. The TagBlock does not contain the payload; it contains only what is needed to locate and verify it.

The TagList aggregates all TagBlocks for a single pinned DAG:

```
// TagList holds all TagBlocks for a single IPFS DAG, ordered by CID,
// and records the root CID of that DAG.
type TagList struct {
    Tags []TagBlock
    Root cid.Cid
}
```

TagBlocks are ordered by the lexicographic ordering of raw CID bytes. This ordering is deterministic: because CIDs are content hashes, the same root always produces the same TagList under the same protocol. TagList size is proportional to the number of nodes in the DAG and independent of payload sizes; §7 quantifies the overhead.

6.4 5.4 Setup: Walking the DAG

Given a root CID and a line.Tagger for the chosen protocol, setup proceeds as follows:

1. Walk the Merkle DAG depth-first starting from the root, collecting every reachable node: root, internal nodes, and leaf blocks. Nodes visited more than once (shared sub-DAGs) are deduplicated by CID. 2. Sort the collected nodes by their raw CID bytes. 3. Present the sorted collection as a blocks.BlockStore with block identifiers set to CID.Bytes(). 4. Pass the store to line.Tagger.TagBlocks, which computes one tag per block and returns them in store order. 5. Pair each tag with its CID to form a TagBlock. Collect all TagBlocks into a TagList with the root CID recorded. 6. Transmit the TagList to the storage node. The client retains the key material required for future challenges: the secret key for Ateniese, SW-Priv, and BJO; the basis for Erway; or simply the public key for SW-Pub.

All DAG nodes are tagged, not only leaf blocks. Internal nodes encode the DAG link structure; a missing internal node makes the subtree it governs unrecoverable regardless of whether its leaf blocks are present. Tagging internal nodes ensures their presence is auditable.

6.5 5.5 Challenge Construction

A challenge targets one or more root CIDs held by a storage node. The verifier selects the roots to challenge, looks up the corresponding TagLists, and concatenates their TagBlocks in a deterministic

order. Because block identifiers are CID bytes and tags are position-independent, the concatenated collection forms a valid block store against which the prover can compute a proof in a single round.

For cross-inventory auditing, where a verifier wants to sample randomly across all content a node holds rather than targeting specific roots, the verifier maintains the full set of TagLists for that node and applies a seed-based selection. For each TagBlock T across all held TagLists, compute:

$$\text{Order}(T, s) = H(\text{CID}(T) \| s)$$

where s is a verifier-chosen random seed. Sort all TagBlocks by Order and take the first N. This produces a challenge set that is deterministic given s and the exact inventory of TagLists, cross-cuts all pinned content simultaneously, and can be independently reproduced by the verifier without transmitting the full block list; only s and N need to be sent.

6.6 5.6 Proof Generation and Verification

Given the challenge set, the storage node constructs a ChallengedList: a blocks.BlockStore backed by live IPFS storage. Block(id) resolves the CID encoded in id and fetches the block bytes from the local IPFS node on demand. The node passes this store and the challenge to line.Prover.Prove, which computes the proof using the underlying protocol's aggregation.

The verifier applies line.Validator.Verify using the key material retained at setup. For Ateniese and SW-Pub, verification evaluates the homomorphic equation over the public key. For Erway, it reconstructs each skip-list path and checks against the basis. For SW-Priv and BJO, it evaluates the inner product using the secret key.

A proof that validates confirms the storage node possessed the challenged blocks at the time of the audit. The statistical guarantee, what fraction of corrupted blocks is detectable with what probability at a given N, is a property of the underlying protocol and is evaluated in §7.

7 6. Security

7.1 6.1 Security Model

The security claim of this construction is bounded and precise. We make no new cryptographic assumptions and introduce no new cryptographic primitives. The security of each protocol variant reduces directly to the security of the underlying PDP or POR construction as established in the original papers surveyed in §3. Our contribution is the mapping layer: the translation of IPFS CID-addressed blocks into the block-store model those protocols expect. The security question for this work is therefore whether that mapping preserves the underlying security properties, specifically whether using CID bytes as block identifiers is a valid instantiation.

We also make a probabilistic claim about detection: a storage node that has discarded or lost a fraction f of its assigned blocks will be caught with measurable probability as a function of the challenge size and total block count. This probability is a property of the underlying protocol's challenge structure, not of any property specific to IPFS.

What this construction does not claim: it does not guarantee security against coordinated multi-node collusion (§4.3, §4.7), which requires economic enforcement mechanisms beyond the scope of the protocol. It does not prevent a server from satisfying challenges by fetching missing blocks on demand; that behavior is addressed by the latency-weighted scoring described in §8.

7.2 6.2 Reduction to Underlying Protocol Security

Each protocol in §3 has a security proof under specific cryptographic assumptions: the RSA assumption for Ateniese and Erway, the computational Diffie-Hellman assumption over BN-256 for SW-Pub, and related assumptions for SW-Priv and BJO. These proofs establish that a computationally bounded prover cannot produce a valid proof for a challenge without possessing the required blocks.

Theorem 1 (CID-Addressed Block Store). Let $\mathcal{S}$ be a BlockStore whose block identifiers are $\{\text{CID.Bytes}(b_i)\}$ for a set of distinct IPFS blocks $\{b_i\}$. Then $\mathcal{S}$ is a valid instantiation of the sparse-identifier block-store parameter in the Ateniese S-PDP, SW-Priv, and SW-Pub protocols, under the collision resistance of SHA-256.

Proof sketch. Using CID bytes as block identifiers is a valid instantiation of the block-ID parameter in each of these protocols. The security proofs for Ateniese, SW-Priv, and SW-Pub treat block identifiers as arbitrary, distinct byte strings; they make no assumptions about how identifiers are distributed or derived. CIDs satisfy the required uniqueness property by the collision resistance of the underlying hash function: two distinct blocks yield distinct CIDs with overwhelming probability. The tag bound to a block via its CID is therefore at least as binding as a tag bound via an arbitrary integer index. The security reductions carry through unchanged.

For Erway and BJO, which use sequential integer positions, the standard sequential instantiation is used. Cross-pin challenges are unavailable for these protocols (§5.1), so no extension of the block-ID model is required.

7.3 6.3 Security of Sparse Identifier Instantiation

Standard PDP and POR formulations index blocks by sequential integers $0, 1, \ldots, N-1$. The IPFS framework substitutes CID byte strings for sequential indices. This section establishes that the substitution preserves the security guarantees of each sparse-capable protocol by tracing the exact mechanism by which block identifiers enter the cryptographic computation.

Ateniese S-PDP. In S-PDP, each block m_i is committed under the tag

$$T_i = \left(h(W_i) \cdot g^{m_i}\right)^d \bmod N$$

where W_i is a per-block binding string. The original paper defines $W_i = V \| \text{encode}(i)$ for a secret random string V and sequential index i; this implementation defines $W_i = V \| id_i$, where id_i is the raw CID byte string of block i.

The Ateniese security reduction (Ateniese et al., 2007, §4) shows that any computationally bounded adversary who produces a valid proof without holding the required blocks can be used to solve the RSA problem modulo N. The reduction exploits exactly one structural property of the W_i values: they must be pairwise distinct, so that the hash images $h(W_i) \in QR_N$ are independently distributed.

Since CIDs are pairwise distinct by the collision resistance of SHA-256, the concatenations $V\|id_i$ are likewise pairwise distinct. The Ateniese reduction therefore applies without modification.

Shacham-Waters Private (SW-Priv). In SW-Priv, block i is committed under the tag

$$\sigma_i = \mathrm{PRF}_K(r_i) + \sum_{j=1}^{S} f_{i,j} \cdot \alpha_j \pmod{P}$$

where r_i is the PRF input bound to block i. This implementation sets $r_i = id_i$, the raw CID byte string of block i, feeding the full identifier directly into the PRF.

The security proof of Shacham and Waters (§3.2, 2008) reduces proof forgery to breaking PRF security. The reduction relies on the joint pseudorandomness of the values $\{\mathrm{PRF}_K(r_i) : 1 \le i \le N\}$, which holds when the inputs $\{r_i\}$ are pairwise distinct. The identifiers $\{id_i\}$ are pairwise distinct by the collision resistance of SHA-256: two distinct blocks yield distinct CIDs with overwhelming probability, so the PRF inputs are pairwise distinct and the security reduction applies unchanged.

Shacham-Waters Public (SW-Pub). In SW-Pub, block i is committed under the tag

$$\sigma_i = \alpha \cdot \left(H(\lambda\|id_i) + \textstyle\sum_j f_{i,j} \cdot u_j\right) \in G_1$$

where H maps byte strings to G_1 via a hash-to-group function and λ is a random 16-byte file identifier. The identifier id_i enters H as raw bytes, with no integer-encoding step. The security proof of Shacham and Waters (§3.3, 2008) requires only that the hash inputs $\lambda\|id_i$ be pairwise distinct, which follows directly from the pairwise distinctness of the CIDs. SW-Pub is inherently identifier-agnostic; no argument beyond CID collision resistance is required.

In all three cases, the sparse structure of the CID identifier space, specifically the absence of any sequential or arithmetic relationship among CID values, is immaterial to the security argument. The reductions require only pairwise distinctness of identifiers, a property guaranteed with overwhelming probability by the collision resistance of SHA-256.

7.4 6.4 Detection Probability

The fundamental probabilistic guarantee is this: a server holding only a $(1 - f)$ fraction of its assigned blocks, having discarded or lost a fraction f, will fail to produce a valid proof for any challenged block it does not hold. If the challenge samples C blocks uniformly at random from a corpus of N blocks, the probability of detecting at least one missing block is:

$$P(f, N, C) = 1 - \binom{N(1-f)}{C} \Big/ \binom{N}{C}$$

This is the hypergeometric detection probability.

Table 2 shows measured detection probabilities from a 100-block corpus with a challenge size of $C = 20$, alongside theoretical values, for each protocol.

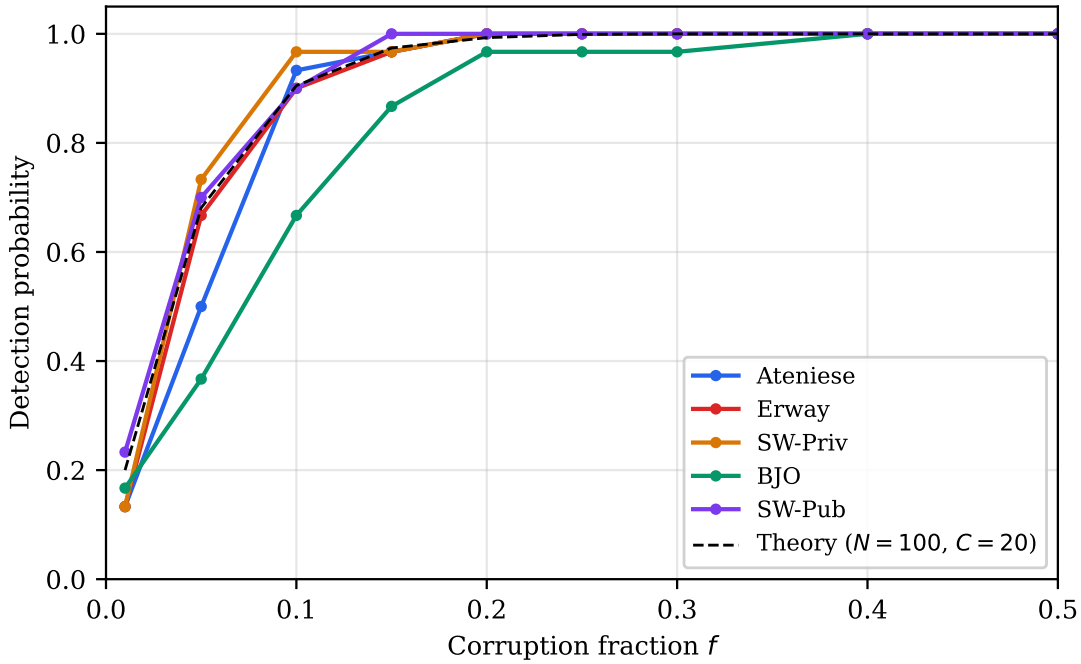

Figure 1: Detection probability vs. corruption fraction f ($N = 100$, $C = 20$). Solid lines: measured; dashed: hypergeometric theory.

Table 2: Detection probability by protocol at varying corruption fractions ($N = 100$, $C = 20$). *Hypergeometric formula.

Corruption (f)	Ateniese	Erway	SW-Priv	BJO	SW-Pub	Theory*
1%	0.133	0.133	0.133	0.167	0.233	0.200
5%	0.500	0.667	0.733	0.367	0.700	0.681
10%	0.933	0.900	0.967	0.667	0.900	0.905
15%	0.967	0.967	0.967	0.867	1.000	0.974
20%	1.000	1.000	1.000	0.967	1.000	0.993

Ateniese, Erway, SW-Priv, and SW-Pub track the hypergeometric curve closely. BJO exhibits substantially lower detection probability at small corruption fractions, a consequence of its inner code structure: challenge responses aggregate blocks through a generator matrix, and the detection probability reflects the code's distance properties rather than the simpler hypergeometric model. At 10% corruption with $C = 20$, the four non-BJO protocols detect corruption with approximately 90–97% probability per challenge round; BJO detects at 67%.

Detection probability scales with challenge size C. For applications requiring higher assurance at low corruption fractions, C can be increased at the cost of proportionally more computation and communication per round. Because challenges are compact (the wire representation of a challenge is

a fixed-size seed plus a count, approximately 41 bytes regardless of C for the seed-based protocols), the primary cost of increasing C is the server's proof computation, not the challenge transmission itself.

7.5 6.5 Resistance to Adversarial Strategies

The detection probability analysis applies directly to subset retention (§4.2): a node retaining only a $(1 - f)$ fraction of its assigned blocks faces detection with probability $P(f, N, C)$ per audit round. Repeated audits compound this probability toward certainty.

On-demand proxying (§4.1), transparent relaying (§4.4), and ephemeral caching (§4.5) are not addressed by proof correctness alone: a node that fetches missing blocks from the network before the deadline can still produce a valid proof. These strategies are distinguishable from honest storage by their response latency; §8 describes how latency-weighted scoring reduces the incentive for these behaviors.

Collaborative fragmentation (§4.3) and sybil-mediated sharding (§4.7) require system-level responses. If a coalition of nodes collectively holds all blocks but no individual node holds its full assigned share, each node will fail its individual challenges at a rate corresponding to its missing fraction. A coordinator tracking per-node challenge outcomes will observe elevated failure rates across the coalition, but dismantling the arrangement requires economic enforcement mechanisms that are outside the scope of the protocol and identified as future work in §8.

Latency-based bypass (§4.6) is addressed directly by the latency penalty described in §8.3.

8 7. Performance Evaluation

The central performance claim of this work is not a comparison among the five protocol variants, as those protocols are established in the literature and their relative characteristics follow from their mathematical structure. The claim is a comparison against the only alternative available to an IPFS client today: downloading the entire dataset and recomputing the Merkle root.

8.1 7.1 Verification Cost: Full Download vs. Remote Proof

To verify that a remote IPFS node is faithfully storing a dataset, a client today must retrieve every block of the Merkle DAG and recompute the root CID from the bottom up. This operation must be performed in full each time the client wants confirmation; Merkle verification provides no state that carries forward to future checks. The communication cost is O(dataset size) per audit. For a multi-terabyte or petabyte-scale archive, this cost is prohibitive: the full corpus must transit the network every time the client wishes to confirm that the storage provider is holding it. Periodic integrity checks at that scale are operationally infeasible, which in practice means they are not performed.

Under the protocol described in this work, an audit round after setup exchanges two messages: a challenge and a proof. For all protocols except Erway, both are constant in size regardless of the dataset. Table 3 shows representative values.

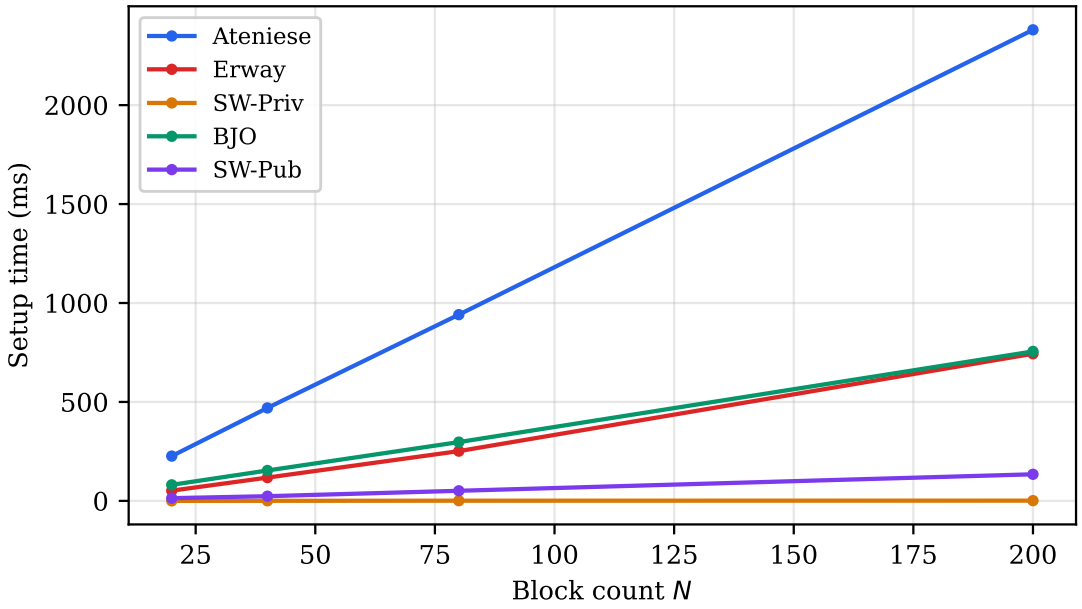

Figure 2: Setup time vs. block count N.

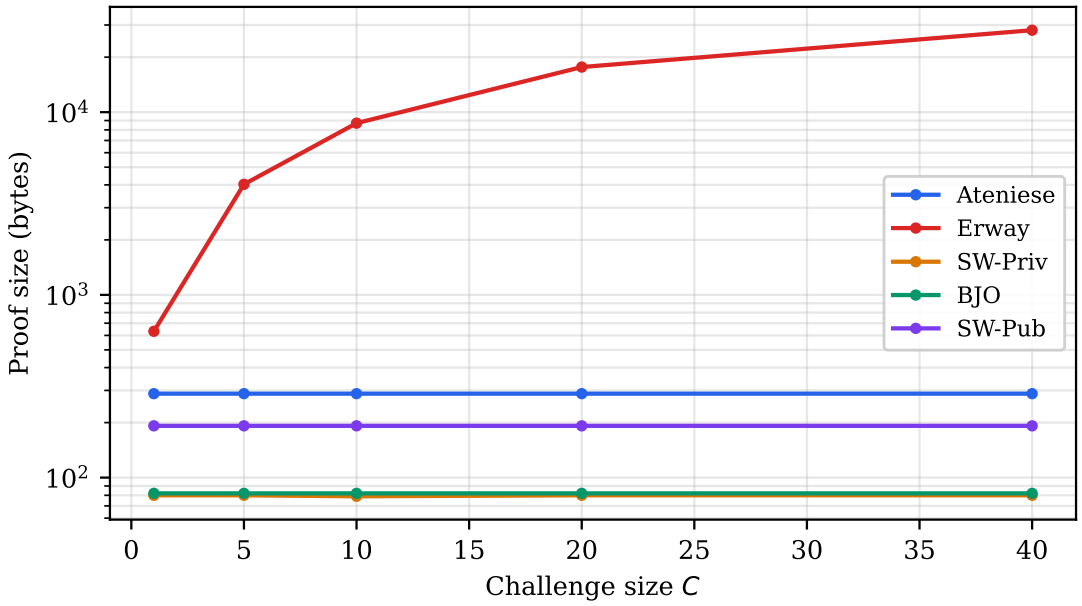

Figure 3: Proof wire size vs. challenge size C.

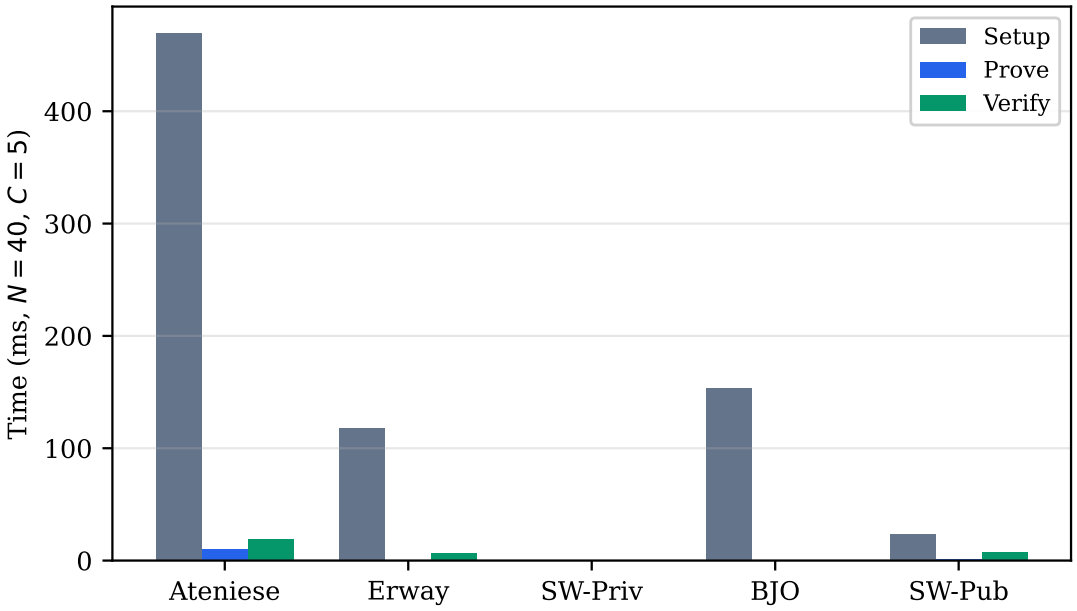

Figure 4: Suite comparison: setup, prove, and verify time at fixed parameters.

Table 3: Verification communication cost: full-download vs. remote proof.

Mechanism	Communication cost
Full-download verify	O(dataset size) — full cost on every audit
Remote proof (constant protocols)	41 B challenge + 80–288 B proof
Remote proof (Erway)	41 B challenge + $O(C \cdot \log N)$ proof

Remote verification is probabilistic rather than certain. A single audit round detects loss of a fraction f of N total blocks with probability $P(f, N, C)$ as given in §6.4. This probability depends on the size of the stored dataset, specifically on N, the number of tagged blocks, and on the challenge size C. For a small dataset (small N), a single challenge may cover a substantial fraction of blocks and yield high detection probability. For a large dataset (large N), the same challenge size C covers a smaller fraction of the total, and $P(f, N, C)$ per round is correspondingly lower. Larger N requires larger C or more frequent rounds to maintain a given assurance level.

What remote verification offers in exchange is that each audit round is independent and its cost is fixed regardless of dataset size. After k rounds, the probability of having detected corruption at least once is:

$$1 - \big(1 - P(f, N, C)\big)^k$$

This cumulative probability increases monotonically toward 1.0 as rounds accumulate. A storage node holding only a partial dataset cannot selectively pass individual rounds, as each round

samples randomly from the full block population, and will eventually fail. The verifier can schedule rounds at whatever frequency the threat model requires; each round costs a few hundred bytes of communication regardless of how large the stored dataset is.

For a petabyte-scale dataset the ratio of per-round remote-proof cost to full-download cost is on the order of 10^{-13}. A client managing such an archive can audit a random cross-section of all pinned content simultaneously once per hour indefinitely at negligible bandwidth cost, while the only alternative, a full Merkle download, would be performed, at best, once.

8.2 7.2 Setup Cost and TagList Overhead

Setup is a one-time operation performed at pin time. The coordinator walks the DAG, computes one tag per node, and stores the resulting TagList. Setup cost scales linearly with block count N and is independent of block content size. Each TagBlock consists of a CID (36 bytes for SHA-256 CIDv1) plus a protocol-specific tag (a cryptographic group element or curve point), in all cases a fixed number of bytes per block, several orders of magnitude smaller than a typical IPFS payload block (256KB under the default Kubo chunker).

The implementations were validated against theoretical performance predictions from the source papers. Table 4 summarizes the key properties of each protocol as implemented; detailed benchmark data is available in the accompanying study.

Table 4: Protocol properties as implemented. *Public verification requires the basis to be published.

Protocol	Sparse IDs	Public verify	Extraction	Proof size	Setup cost
Ateniese S-PDP	Yes	No	No	Constant	High (RSA)
Erway D-PDP I	No	Yes*	No	$O(C \cdot \log N)$	High (RSA)
SW-Priv	Yes	No	Yes	Constant	Low
BJO	No	No	Yes	Constant	Moderate
SW-Pub	Yes	Yes	Yes	Constant	Moderate

RSA-based schemes (Ateniese, Erway) require a security parameter $k \geq 1024$ bits for production use, and setup cost scales steeply with k. The pairing-based and PRF-based schemes (SW-Pub, SW-Priv) do not carry this constraint and their setup cost at production parameters is representative of test measurements.

8.3 7.3 Detection Probability

The challenge size C is a configurable parameter that governs the trade-off between audit cost and detection probability. Table 2 (§6.4) shows measured detection probabilities at $C = 20$ for a 100-block corpus. At 10% corruption, four of the five protocols detect loss with 90–97% probability per round; repeated audits drive detection probability toward certainty. C can be increased to raise assurance at lower corruption fractions; because the challenge is a compact seed, the primary cost of increasing C is server-side proof computation rather than communication.

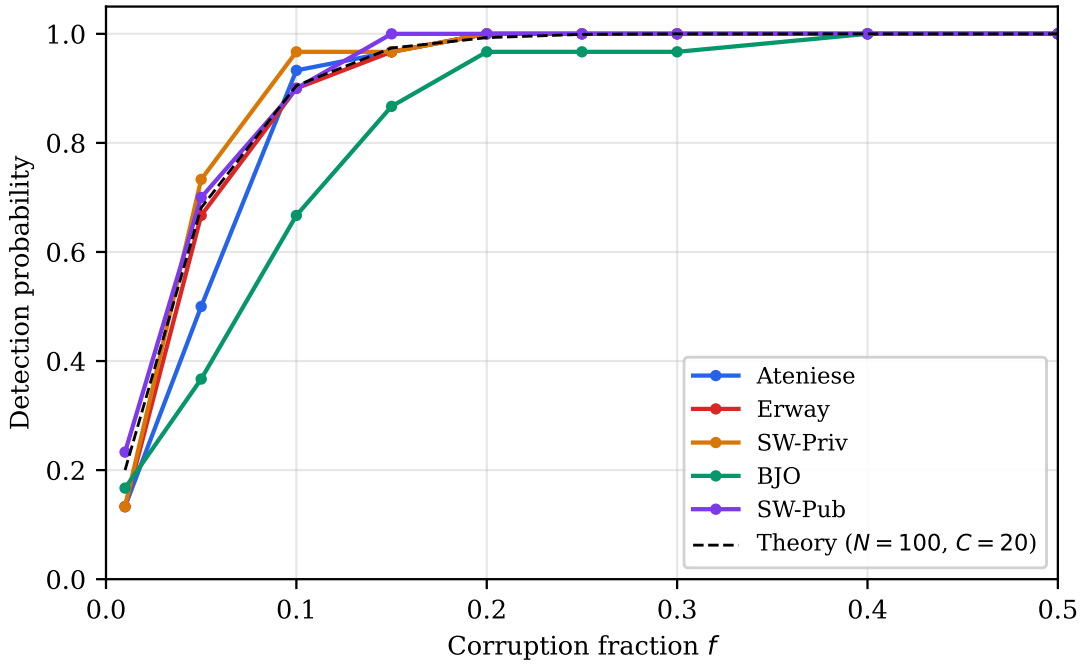

Figure 5: Detection probability vs. corruption fraction f ($N = 100$, $C = 20$). Solid lines: measured; dashed: hypergeometric theory.

9 8. Discussion

9.1 8.1 Protocol Selection Trade-offs in Practice

The five protocol variants differ along three dimensions that have direct operational consequences in the IPFS setting: support for sparse block identifiers, the ability to perform public verification, and the provision of an extractor.

Support for sparse block identifiers, meaning acceptance of arbitrary byte strings as block identities rather than requiring sequential integer positions, is required for cross-pin challenges. Only Ateniese, SW-Priv, and SW-Pub satisfy this requirement. Erway and BJO are restricted to auditing a single sequentially-ordered TagList per round. A deployment that requires cross-inventory auditing across multiple pinned datasets must therefore choose from the sparse-identifier group.

Public verification changes the trust model for the coordinator role. With SW-Pub, the client publishes a public key at setup and may thereafter delegate auditing to any third party without sharing secret material. This enables scenarios in which a neutral auditor, neither the client nor the storage provider, continuously monitors storage fidelity and reports results. With Erway, public verification is possible if the client publishes the basis; the basis carries no secret material and its publication is a deliberate policy choice. The remaining protocols, Ateniese, SW-Priv, and BJO, require the auditing party to hold secret key material, restricting the auditor role to the original client.

Extraction determines whether the protocol can serve as a recovery mechanism in addition to a detection mechanism. SW-Priv, BJO, and SW-Pub provide extractors: a party that issues enough challenges and collects the responses can reconstruct the full dataset by solving a system of linear equations over the underlying field, without direct access to the server. Ateniese and Erway provide possession guarantees only; a server that has lost data is detectable but the data is not recoverable through the audit protocol.

Setup cost separates the RSA-based schemes from the pairing- and PRF-based ones. Ateniese and Erway require generating RSA parameters at a security parameter $k \geq 1024$ bits and computing one modular exponentiation per block at setup; this cost dominates for large datasets. SW-Priv requires only field arithmetic, making it far cheaper at setup. SW-Pub requires elliptic-curve point multiplication per block at BN-256 parameters, placing it between the two extremes. BJO includes an outer ECC encoding pass in addition to tag computation.

No single protocol is dominant across all dimensions. The trade-offs among them are determined by the verification trust model, the requirement for cross-pin challenges, the recoverability requirement, and the tolerance for setup cost at scale.

9.2 8.2 Reputation and Confidence Scoring

A single challenge round produces a binary outcome: the proof is valid or it is not. The statistical value of this outcome depends on N, C, and the unknown corruption fraction f. For a large dataset with a small challenge, a passing round provides modest assurance per round; for a small dataset with a large relative challenge, a passing round provides high assurance. Neither outcome alone supports a strong statement about storage fidelity.

A coordinator that maintains a history of challenge outcomes for each storage node can aggregate these outcomes into a confidence score. After k successful audit rounds, the probability that a node retaining only a $(1 - f)$ fraction of its assigned blocks has passed all k rounds by chance is:

$$\left(1 - P(f, N, C)\right)^k$$

For fixed f, N, and C, this probability decreases geometrically in k. A coordinator tracking this quantity accumulates increasing statistical confidence that a node is storing the assigned content, or conversely, an increasing probability of having detected corruption if the node has ever failed a round.

This framework also supports per-node reputation modeling. Two nodes that have each passed 100 rounds are not necessarily equally trustworthy: if one manages a dataset ten times the size of the other at the same challenge size C, its per-round detection probability is lower and its cumulative assurance grows more slowly. A reputation system that scores nodes on raw pass rate treats these cases equivalently; a system that tracks the underlying probability mass assigned to each passed round does not.

9.3 8.3 Latency-Weighted Scoring

A proof that is cryptographically valid but arrives after a delay that is inconsistent with local storage is evidence of on-demand retrieval, not local possession. The adversarial strategies of §4, on-demand proxying, transparent relaying, ephemeral caching, and latency-based bypass, share the property that the storage node must fetch absent content from the network before responding. Network propagation adds latency; locally held data does not.

Integrating response latency into the audit scoring model creates an economic disincentive for these strategies. A coordinator that records the round-trip time of each challenge-response pair and penalizes responses above a threshold latency reduces the reward value of proofs produced via on-demand fetching. The threshold must be calibrated to the expected distribution of legitimate response latencies, which varies with geographic placement, hardware, and dataset size; a threshold set too aggressively will penalize honest nodes on high-latency links, while one set too loosely will fail to distinguish proxying from possession.

Latency measurement must be treated as advisory rather than dispositive. A latency penalty degrades the effective reputation score of a node exhibiting fetch-consistent latency; it does not by itself constitute proof of non-compliance. The appropriate system-level response to sustained high-latency audit responses is a reduction in confidence score and, eventually, contract termination or dispute initiation, not a binary pass/fail determination based on a single round's timing.

9.4 8.4 Toward a Trustless Federated Pinning Network

The protocol described in this work is a necessary but not sufficient condition for a trustless federated pinning network. It provides the verification primitive: given a TagList for a pinned dataset, a coordinator can confirm that a remote storage node held a random cross-section of that dataset at a specific point in time, by exchanging a few hundred bytes. What the protocol does not provide is an economic enforcement layer: a mechanism to translate detection of non-compliance into consequences for the non-compliant node.

A complete trustless storage network would combine the audit protocol of this work with at minimum the following components. A staking mechanism by which storage nodes commit collateral that can be slashed upon proven non-compliance. A dispute protocol by which a coordinator can produce a cryptographic record of a failed audit round, comprising a valid challenge alongside the server's invalid or absent proof, and submit it to an adjudicator. A payment protocol by which storage fees are released to the storage node contingent on audit performance, rather than in advance. These components interact: the value of the stake must exceed the economic gain from defection, the dispute protocol must be efficient enough that pursuing it is worthwhile, and the payment schedule must not create incentives for nodes to defect at end-of-contract.

The design of incentive-compatible mechanisms for federated pinning networks draws on the economics of rational agent behavior in decentralized systems and is a distinct research problem from the cryptographic verification problem addressed here. We identify it as the principal direction for future work.

9.5 8.5 Open Questions

Several questions are left open by this work.

Collusion resistance. Collaborative fragmentation (§4.3) and sybil-mediated sharding (§4.7) are detectable at the per-node level, as each node in the coalition will exhibit elevated challenge failure rates proportional to its missing fraction, but the coordinator cannot easily distinguish a colluding coalition from independent nodes each experiencing partial storage loss. Distinguishing correlated failure patterns indicative of coordination from uncorrelated partial failure requires modeling the joint distribution of outcomes across nodes, which is complicated by the variability of detection probability with dataset size and challenge parameters.

Multi-provider scenarios. A client that requires k-of-n availability guarantees, meaning any k of n storage providers can reconstruct the dataset, must combine the audit protocol with a redundancy scheme, such as erasure coding across providers. The interaction between the redundancy structure and the TagList model, particularly for protocols with extractors, is not addressed here. Cross-provider auditing, in which a single challenge round samples from the aggregate inventory across multiple providers, is a natural extension of the cross-pin challenge construction of §5.5 but requires coordination infrastructure not described in this work.

Parameter selection. The challenge size C is the primary tunable parameter governing the assurance-versus-cost trade-off. Selecting C optimally for a given dataset size N, a target cumulative detection probability after k rounds, and a specified corruption fraction f of concern is a constrained optimization problem. A practical deployment system should expose this parameter selection to the coordinator rather than requiring manual tuning, and should update C recommendations as N changes over the dataset's lifetime.

10 9. Conclusion

IPFS provides strong integrity guarantees at retrieval time but no protocol-level mechanism for verifying that a remote node is currently storing assigned content. Pinning services, the persistence layer that makes IPFS viable for archival use, operate entirely on trust. The only means a client has today to confirm that a storage provider is faithfully retaining a dataset is to retrieve and reconstruct it, a cost equal to the full dataset size, incurred anew on every verification.

This paper establishes that CID-addressed Merkle DAG storage is a valid instantiation of the sparse-identifier block-store model required by PDP/POR protocols (Theorem 1), and provides a general framework, realized by the TagBlock/TagList abstraction, through which the Ateniese, SW, and BJO protocol families can be applied to IPFS pinning without modification to their cryptographic cores. The five protocol families serve as case studies validating the framework across the key design axes; the contribution is the mapping layer and the proof of its soundness. This mapping enables audit rounds that exchange a fixed-size challenge and proof regardless of dataset size, replacing O(dataset size) per-verification cost with a constant-sized round-trip. A petabyte-scale archive can be audited by exchanging a few hundred bytes. The same mapping enables cross-pin challenges: a single compact challenge can sample uniformly across all content pinned to a storage node simultaneously, auditing the node's full storage commitment in one round.

We define the TagBlock and TagList data structures that make this concrete, describe the DAG

traversal procedure that produces them at setup, and implement five protocol variants spanning the relevant axes of the protocol space: static and dynamic data, private and public verification, possession-only and retrievability guarantees. The security of each construction inherits directly from the security of the underlying protocol; no new cryptographic assumptions are introduced.

Remote verification is probabilistic. A single round detects loss of a fraction f of N blocks with probability $P(f, N, C)$ determined by the challenge size C and the corpus size N. This probability is lower than certainty, but each round is independent and its cost is fixed. Cumulative detection probability after k rounds follows $1 - (1 - P(f, N, C))^k$ and increases monotonically toward 1.0. A storage node that has silently discarded data cannot indefinitely pass random challenges drawn from the full block population. Certainty is approached asymptotically rather than achieved in a single round, but it is approached at negligible marginal cost per round.

The outstanding problem is the economic enforcement layer: the staking, dispute, and payment mechanisms that translate a detected integrity failure into consequences for the non-compliant node. Without that layer, the audit protocol is a monitoring tool rather than a trustless contract. Its design is identified as the principal direction for future work.