EigenDA: The Hyperscale Verifiable Data Availability Layer

Eigen Labs

Abstract

Data availability (DA) has become the hidden bottleneck of modular blockchains: the faster the consensus layer grows, the more fragile its access to verifiable data becomes. EigenDA introduces a cryptoeconomically secure DA protocol that decomposes this bottleneck into three distinct layers: storage, access, and observability. Each layer is optimized for a different trust and performance regime. At the *storage layer*, validators are delegated with stake through the EigenLayer protocol to provide Byzantine Fault Tolerant (BFT) guarantees using the Verifiable Information Dispersal (VID) scheme [13]. The *access layer* scales elastically by embedding economic incentives into CDN-grade infrastructure, yielding elastic capacity, high throughput, and resilience to denial-of-service attacks. Cryptoeconomic security arises from the *observability layer*, where light-node observations inform local judgments that coordinate social consensus, with outcomes enforced via intersubjective forking on EigenLayer [12]. Through formal analysis of our Data Availability Sampling (DAS) protocol, we derive finite-sample guarantees for observation soundness and show that observability can be converted into cryptoeconomic security via intersubjective forking. Together, these mechanisms form a practical DA layer that provides a verifiable substrate for rollups, verifiable services, and sovereign AI agents—systems where any lapse in availability equates to a collapse of safety. This white paper demonstrates a complete protocol design that specifies the job of each layer and the fee and incentive flows, with explicit assumptions and clear guarantees for throughput and cryptoeconomic security.

1 Introduction

Data availability (DA) has emerged as one of the core primitives that power the modern generation of modular and scalable blockchain systems. Data availability is especially important in the context of the so-called "Layer 2" mechanisms [7] that augment the speed, expressivity, or efficiency of the blockchain itself. For instance, optimistic and ZK rollups provide Layer 2 mechanisms for verifying the correctness of state machine transitions that can scale to extremely high throughput due to their much higher computational efficiency. However, for optimistic rollups to remain secure and ZK rollups to remain live [1], they must in general make their transaction data (or other information needed to reconstruct the state) available, so that the watcher nodes can verify the rollup's correctness and end users can reconstruct and validate their own account state.

Beyond Layer 2 rollups, EigenCloud Autonomous Verifiable Services (AVSs) [6] constitute another foundational category of data availability consumers. AVSs generalize the functionality of Layer 2 rollups into a much broader domain of verifiable applications such as verifiable AI inference services and sovereign AI agents. Given the data-intensive nature of these more expressive application types, AVSs promise to dramatically increase the demands for data availability throughput.

[1] Even when state transitions are proven correct, a ZK rollup still requires data availability so the post-state can be reconstructed by anyone and a new sequencer can build valid blocks on the tip; without it, liveness can fail [9].

Critically, for most of these systems, data availability failures can translate directly into safety failures for the consumer application. For instance, if the malicious sequencer for an optimistic rollup is able to corrupt data availability, they may also corrupt the rollup's Layer 1 (L1) bridge by posting an incorrect state transition which watcher nodes will be unable to challenge [8]. Thus, verifiable applications that rely on a data availability service generally require high security from that service in order to prevent it from presenting a security risk.

Given these needs, a satisfactory data availability system must address three critical areas: providing the highest possible grade of security, achieving practically unbounded scale, and minimizing operational costs. We present the EigenDA protocol, a DA protocol designed to address these areas.

The key innovations of EigenDA are as follows. First, we introduce a first-principles architectural factoring that cleanly separates storage, access, and observability layers (Section 1.1). Second, we develop a highly scalable and verifiable storage scheme based on erasure coding and stake-weighted attestations (Section 1.2). Third, we present an adversarially-robust access mechanism that combines elastic capacity with economic incentives to resist denial of service attacks (Section 1.3). Fourth, we design a fully robust and scalable observability scheme that provides cryptoeconomic security through Data Availability Sampling (DAS) and intersubjective slashing (Section 1.4). We discuss each of these contributions in the following sections.

1.1 Architectural Principles

The EigenDA architecture is founded on a set of architectural principles which emerge from a joint consideration of security and performance. Below, we provide an intuitive overview and motivation for these principles.

The *first principle* relates to the division between storage and access of data. In the EigenDA protocol, storage and access are separate functionalities which are performed by distinct architectural layers. The storage layer (Section 1.2) is responsible for providing a durable availability guarantee to the access layer, while the access layer (Section 1.3) is responsible for making data accessible to a large number of end users.

The essential reason for this separation is that, while storage and access are both necessary components of secure data availability, storage can generally only be (efficiently) provisioned in a threshold honest trust setting, while access is amenable to a weaker "any trust" model:

- "Threshold Trust" Storage: The requirement of storage means that data must be persistently stored by nodes of a network before the protocol can certify the availability of any data. Various tradeoffs will generally push this storage function toward a threshold trust model. For instance, in order to provide strong guarantees of liveness, the system will have to produce a certificate of availability even when some nodes have not attested to storing data.

- "Any Trust" Access: Given a durable storage layer, an access layer can operate in a much weaker "any trust" model, which requires only that at least one honest full node exists (see details in Section 4.1). As long as there is a single honest full node which is able to relay data from the storage layer to end users, accessibility of the data is guaranteed. This assumption is quite weak indeed, since even in the event that 100% of the existing collection of full nodes is corrupted, various measures can be taken to update the configuration of the access layer in order to ensure the release of data.

Given these different trust models, specialization of the storage and access layers is an effective way to jointly maximize security and performance. On the one hand, the threshold trust assumption of the storage layer is strengthened when this layer is highly decentralized. On the other hand, given its weaker trust assumption, the security of the access layer is less dependent on strong decentralization. As we will discuss later, the functionality of access itself can require specialized capabilities and infrastructure in order to meet the demanding requirements of the access task (especially in adversarial modes). Architecturally separating these layers allows the protocol to maintain lightweight requirements for the storage layer that are amenable to decentralization while allowing the access layer to optimize heavily for performance and scale.

From a security standpoint, the "any-trust" model used in the access layer is quite strong: because the access layer is ephemeral, it can be hot-swapped in the event of an attack. Recall that this same guarantee does *not* apply to the storage layer. Storage relies on a weaker threshold-trust model, which by itself is insufficient for applications that demand a high level of security. For such applications, we must ensure that any successful attack on the storage layer would impose an economic penalty that exceeds any potential gain to the attacker.

This requirement motivates our *second principle*, which introduces the observability layer. As we discuss in Section 1.4, achieving strong economic security depends on extending the DA protocol so that members of the broader community can reliably detect, with high accuracy, when a DA attack has occurred. Architecturally, this necessity gives rise to a distinct third layer: the *observability layer*. To avoid adding load to the storage layer and to strengthen the protocol's ability to detect attacks, the observability layer interfaces solely with the access layer.

The principles described above lead to a system architecture which separates the storage, access and observability layers, as shown in Figure 1.

1.2 Scalable Storage

EigenDA makes use of verifiable erasure coding in order to distribute encoded chunks among validators. We use the term *blob* for a data object stored by the DA system. Each blob is erasure-coded into chunks such that any sufficiently large subset of chunks can reconstruct the original blob. The validators are backed up by stake through the EigenLayer protocol [12]. Under the chunk-assignment scheme (Section B), each validator receives a number of chunks proportional to its stake. Because assignment and attestations are stake-weighted, the scheme attains BFT availability under standard assumptions (see Section 5.1 for analysis).

1.3 Adversarially-Robust Access

An important aspect of data availability systems is that they are often used to support an "any-trust" security model for downstream applications. For example, an optimistic rollup may depend for security on the property that any of its validators are able to download all data from the data availability system. The rollup achieves strong security guarantees when this validator set is open and permissionless.

The fact that honest consumers of data are a permissionless set presents an immediate challenge for data availability protocols. On the one hand, for each item of data, the demand from honest consumers (who may or may not trust each other) to download this item is infinite. But more importantly, the problem framing inherently prevents the DA protocol from distinguishing between

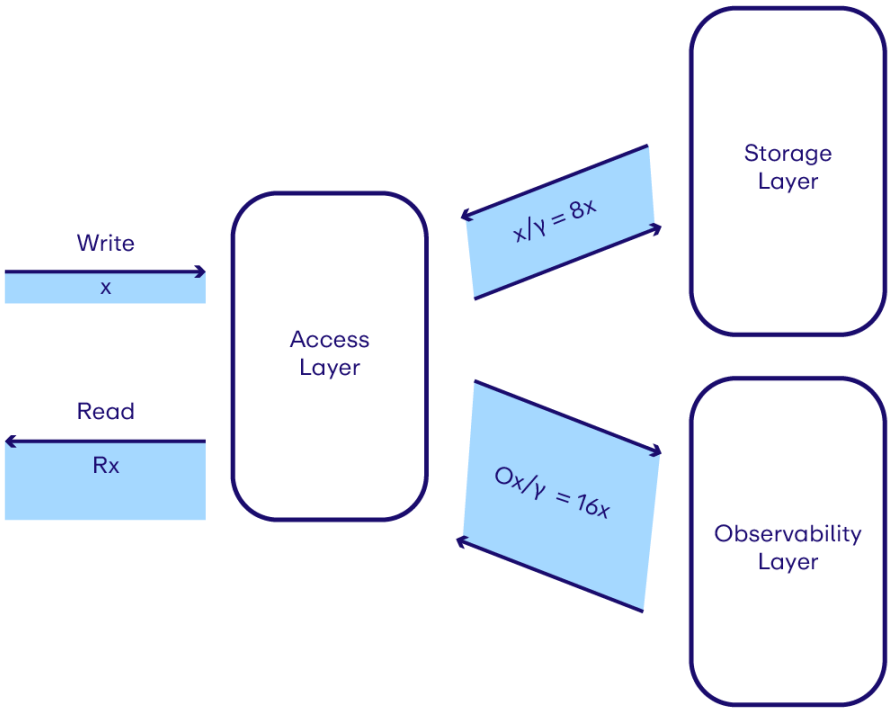

Figure 1: Heterogeneous layers of system architecture. The access layer accepts write requests at an effective data throughput x. The associated blobs can be read many times from the access layer, resulting in an effective read amplification R. Encoding adds redundancy of magnitude $1/\gamma$ to the chunks written to the storage layer. These chunks can also be read back by the access layer at a rate of x/γ per access layer node. Finally, the stochastic assignment scheme of the Data Availability Sampling (DAS) protocol adds overhead of O (typically between 2 and 5) to the observability layer's data write rate when DAS observation is taking place. Chunks can be read back from the observability layer to the access layer at a rate of x/γ per access layer node, similarly to the storage layer.

honest consumers who are retrieving data for valid purposes and malicious consumers who are retrieving data merely to consume resources and thereby deny service to honest parties.

Most protocols do not robustly address the denial of service risk. We illustrate this via a sequence of straw men:

1. *Bring your own bandwidth (BYOB).* Many existing protocols implicitly address the denial of service concern by recruiting the help of consumers themselves in a "bring your own bandwidth" (BYOB) scheme: Users wanting to retrieve data from the network must do so by joining a peer-to-peer (P2P) network which asks them to contribute their own bandwidth in order to serve that same data to other users in the P2P network. While conceptually elegant, the BYOB paradigm depends on heuristics for filtering out adversarial network participants and tends to be susceptible to sophisticated P2P attacks or crude Sybil attacks which can compromise the actual availability of data.

2. *Pay for retrieval.* Another possible approach is to require users to pay a fixed price for retrieval from the protocol, an approach which is exemplified by the recently unveiled Shelby [10] protocol. However, while this approach helps incentivize high performance from data providers, it doesn't inherently address the deeper concerns around denial of service, since it may still be relatively inexpensive for an attacker to pay the cost of exhausting the full upload capacity of the network.

3. *Retrieval auction.* Retrieval payments can be made more effective at deterring denial of service attacks by using an auction mechanism that raises retrieval prices during periods of congestion, thereby increasing the cost for an adversary to sustain an attack. However, the strength of this protection fundamentally depends on the system's upload capacity. We can define a factor, denoted F, that measures how much more an attacker must spend compared to honest users. This factor is given by the total upload capacity of the data availability (DA) system divided by the total download demand of honest consumers. For example, suppose the system has enough capacity to serve each blob 100 times, and there are 10 honest consumers who each download the blob once. In this case, the factor F is 100 divided by 10, which equals 10, so the best the system can do is to force an adversary to spend 10 times the honest users' total retrieval budget in order to deny them service. This example shows that simply adding a retrieval auction to a system that has not otherwise been designed to maximize this factor provides only limited protection. It leads to an unfavorable trade off: either honest users must maintain a very large retrieval budget, or the cost of carrying out a denial of service attack can remain small compared to the potential gains from a successful attack.

4. *Elastic capacity.* In the web2 world, distributed denial of service (DDoS) attacks are often mitigated via the use of content delivery networks (CDN) consisting of heavy duty distributed caching layers that serve data from the edge at reduced cost. In practice, many CDNs have a net delivery capacity exceeding the attack capacity of even highly determined attackers [4]. From the standpoint of a decentralized DA protocol component making proper use of a CDN, the CDN can be viewed abstractly as an ability to scale upload capacity to any arbitrary level needed to serve even adversarial demands. While such an approach can be effectively used to ensure that no request, honest or adversarial, goes unserved, the cost to the protocol actor can be substantial in the case of well-equipped adversaries.

While none of these straw men are individually sufficient to address the denial of service attack in a satisfactory way, we find that by combining them together we can achieve a solution that elegantly addresses all areas of concern. This solution takes the form of a scalable delivery capacity (e.g., via the use of CDNs) combined with retrieval auctions. In the idealized scenario that a CDN can always scale beyond adversarial attack capabilities, this approach renders all attacks completely null and void, since all honest users retain the ability to retrieve data at a baseline fee and no net cost is incurred by the protocol operators. Otherwise, this model falls back to a retrieval auction model described above with very large F, making the cost of the DDoS attack formidable.

1.4 Economic Security and Observability

In general blockchain parlance, the idea of economic security refers to the guarantee that a defined amount of economic penalties will be incurred by any party that successfully attacks the protocol in a way that causes one of its security guarantees to fail. Generally, this guarantee is furnished in a layered manner. The first layer consists of a Byzantine Fault Tolerant (BFT) protocol executed by the validators which ensures that all security properties will hold as long as a sufficiently large contingent of the validators properly executes the protocol. The second layer consists of an explicit or implicit rule that, in the event that the BFT security assumption fails, a new instance, or "fork" of the protocol will be created, in which all validators participating in the malicious behavior of the previous fork will have incurred an economic penalty.

An implicit requirement of this layered model of security is the notion of public verifiability of the protocol logic [12]. For instance, if the Ethereum blockchain were to accept an incorrect state transition due to malicious activity of the validators, the incorrectness of the state transition could be easily demonstrated to members of the public who could then take the rational action of migrating to the new protocol which is now maintained only by validators that are not demonstrably corrupt.

When it comes to the security property of data availability, achieving this same level of verifiability comes with certain intrinsic challenges. These challenges are well demonstrated by the "Fisherman's dilemma" [2] which points to the essentially subjective nature of data availability. State transition correctness is a fully objective property of a chain, in the sense that after checking the correctness of a state transition, an observer can be fully confident that anyone else who performs the same check will reach the same conclusion. However, the private nature of data availability does not lend this same simplicity: A DA provider might serve data to one observer but not another; and vice versa; my observation by itself tells me nothing about the global/objective availability of the data.

Equipping a DA protocol with economic security thus requires consideration of the related problem of verifiability. Naturally, the gold standard for addressing this problem would be to achieve a fully objective grade of availability. In such a world, any observer making a positive observation that data was available would imply that all end users could successfully retrieve data, and vice versa. While data availability sampling (DAS) claims a solution for this, we will argue that most existing actual and theoretical instantiations of DAS rely on assumptions that are unsatisfiable with current networking technology [14].

Given this observation, it is natural to adopt a generalized notion of verifiability for data availability, in which observers collectively assess whether a data item is available subject to bounded error. We refer to this notion as *observability* and formalize it below.

While observability is a strictly weaker notion than verifiability, we note that it is sufficient as

a primitive for supporting economic security. Since economic security itself is a broad community and market-driven function of forking a chain in the event of a collectively determined safety failure, the important question is not whether an individual observer has access to a strictly sound measurement of availability, but rather whether the community at large is able to aggregate its collection of measurements into a sound judgment. The EigenDA protocol aims for a small false positive fraction (the fraction of honest light nodes that consider data as available when it is actually unavailable) and a negligible false negative fraction (the fraction of honest light nodes that treat data as unavailable when it is actually available). The protocol assumes that this bounded noise at the level of individual observations can be filtered out by the larger community. The relationship between the false positive and false negative fractions and the system parameters is discussed in Section 5.2.

Observability, in the sense defined here, is not a de facto property of most existing DA protocols. It can also be in tension with other priorities such as scalability. To make this concrete, we present a series of straw men that appear, explicitly or implicitly, in prior formal and informal treatments of DA verifiability and DAS.

1. *Observers download blobs.* Check whether data is available by checking if I can download an entire blob. This approach is neither scalable nor sound. It is not scalable because in order to observe whether data is available, an observer needs to be able to download the full traffic of the data availability system. This is a large cost both to observers and to the DA network. It's also not sound since a blob available to one node may not be available to others.

2. *Observers download blobs and share them.* The previous scheme can be made sound at the cost of further reducing its scalability. When an honest observer downloads a blob, it must then serve that blob to all others who might want it, whether observers or end users. This is a much greater cost to the observer, and requires a dynamic scale of capacity that is unlikely to be achievable except at small scales of throughput.

3. *Observers download samples.* The central idea of data availability sampling is to increase scalability by having each observer fetch only a small set of samples per blob. However, naively downloading samples yields weak guarantees: an adversary can serve only the sampled chunks to a specific observer while withholding others. In the worst case, this provides no assurance of availability. In the best case, a well-resourced user could emulate many observers to obtain a stronger guarantee. This scheme is used in Celestia (see more details in Section 2).

4. *Observers download samples via obfuscation network.* The original DAS protocol provides strong guarantees to individual nodes under an anonymity assumption: requests for individual samples are indistinguishable, so a data provider cannot link multiple sample requests to a single node [14]. A straightforward way to approximate this assumption is to route queries through an obfuscation network; however, current approaches remain preliminary and encounter scalability challenges.

5. *Observers download samples and reshare them.* This approach can achieve partial observability (a bounded false positive fraction), but not verifiability, due to the possibility of a collection of nodes being deceived by a sophisticated adversary. The false negative fraction of the protocol can be high in absence of a reliable reconstruction mechanism. On the other hand, this approach has limited scalability due to the requirement of each observer to upload

each of its sampled chunks to a potentially large set of other observers. This scheme is used in PeerDAS (see more details in Section 2).

Our contribution in this paper is to specify a DAS protocol which achieves the formal properties of observability in a rigorous manner without sacrificing scalability. We do this by providing an architecture that gives a concrete foundation for our DAS protocol: EigenDA introduces a set of high-resource full nodes that recover and serve data in the DAS protocol, with security derived from an explicit existential honesty assumption for these full nodes. This architecture allows us to establish formal security guarantees without relying on the heuristics about P2P networks. We also formalize the attacker's game in DAS and provide an analysis for the recoverability properties of a collection of randomly sampling observers under throughput constraints.

1.5 Overview of This Paper

The remainder of this paper provides a comprehensive technical description of the EigenDA protocol.

Section 2 outlines existing approaches to data availability.

Section 3 introduces a formal model for Observable Data Availability (ODA) schemes, defining the key operations and properties that characterize secure data availability systems.

Section 4 presents the design of the EigenDA protocol, including our architectural principles, system components, cryptographic primitives, and detailed protocol operations.

Section 5 provides a rigorous security analysis, demonstrating EigenDA's safety and liveness properties under Byzantine conditions and analyzing the observability guarantees of our DAS protocol.

Section 6 discusses key design decisions and their implications, including our approach to denial-of-service resistance and comparisons with alternative DAS designs.

The appendices provide additional technical details, including formal proofs of our security properties and concrete instantiations of our cryptographic schemes.

2 Related Works

Foundational work on asynchronous verifiable information dispersal (AVID) [3] presents an efficient protocol that ensures consistent, retrievable storage under Byzantine faults with near-optimal communication and storage overhead. Later works refine this line of research: AVID-FP [11] reduces AVID's communication costs via homomorphic fingerprinting while preserving chunk verifiability and retrievability, whereas AVID-M [19] further improves communication efficiency using Merkle-tree-based fingerprints but sacrifices verifiability, which causes retrievability to fail under malicious encoding. More recent work, semi-AVID-PR [13], introduces a communication-and-storage-efficient verifiable information dispersal protocol that ensures provable retrievability for rollups using linear erasure coding and homomorphic vector commitments, while avoiding the heavier termination guarantees of prior AVID schemes.

Several recent works build on the foundation of verifiable information dispersal by improving the performance of the decentralized network for tasks such as recovery of missing data shards. For instance, Walrus [5] introduces a novel two-dimensional erasure-coding scheme to enable efficient data shard recovery when validators exit or join the network, and Shelby [10] utilizes Clay codes [17] to reduce the networking overhead in the event of reconstruction.

These variations to the VID paradigm optimize for specific performance characteristics. The performance improvements come with different design trade-offs: alternative encoding structures may use different cryptographic primitives than polynomial commitments with succinct opening proofs, and encoding schemes that reduce redundancy (as in Shelby) correspondingly affect Byzantine fault tolerance thresholds according to standard erasure coding analysis.

The access and observability models in Shelby and Walrus differ from those explored in EigenDA. Shelby's design emphasizes paid retrieval as a strategy for incentivizing high-throughput retrieval, and introduces an auditing scheme that makes storing data incentive compatible within its system. These mechanisms provide robustness measures suitable for certain application contexts. In contrast, EigenDA's access and observability protocols are specifically designed for adversarial scenarios where significant economic value (such as funds locked in rollups) may create strong attack incentives, which represents a different threat model and corresponding set of security requirements.

Some existing protocols, such as Celestia [1], have made a compelling effort to address the problem of observability via the first fielded DAS protocols. However, as of yet, none of these protocols currently jointly address the problems of scalable storage and distribution; moreover, as noted above, existing DAS deployments tend not to address important problems such as data recovery, and also provide analyses based on hidden assumptions which are not realized in practice (see Section 6.1).

An upcoming Ethereum upgrade introduces a data availability sampling protocol known as *PeerDAS* [16]. PeerDAS employs data availability sampling to scale data storage within the Ethereum protocol, taking a different architectural approach than VID-based dispersal schemes. The protocol incorporates samplers re-sharing their samples (an approach discussed as Straw man #5 in Section 1.4), reflecting design choices optimized for Ethereum's specific deployment context and existing peer-to-peer network infrastructure.

3 Definitions and Modeling

In this section, we provide a formal model for the ODA scheme and define the key properties of a DAS system.

3.1 Observable Data Availability Protocol

To reason formally about the protocol, we introduce a formal model of an **Observable Data Availability (ODA) scheme**. This model captures the essential operations and properties needed for a decentralized data availability system that can scale while maintaining strong security guarantees. An ODA scheme consists of four core operations that capture how data is handled and verified in the system.

Definition 3.1 (Observable Data Availability Protocol)**.** *An observable data availability protocol has four procedures:*

- `Disperse`(**blob**) $\rightarrow$ **cert** – *This operation takes an input data blob and disperses it to the network, often by distributing encoded chunks to a collection of validators, and returns a* ***data availability certificate*** *for that blob. The certificate can be used by others to reference and verify the blob's availability. Generally, a certificate will consist of a commitment to the dispersed blob that has been cryptographically signed by staked validators in the network.*

- `Verify`(**cert**) $\rightarrow$ {**true**, **false**} – *Verification is a deterministic check that returns* ***true*** *if and only if the certificate is valid according to the rules specified by the specific DA system, indicating that the blob is supposed to be properly stored.*
- `Retrieve`(**cert**) $\rightarrow$ **blob** – *This operation attempts to retrieve the original blob using the information in the certificate, e.g., by contacting storage components in the network and attempting to download the blob or recover it from encoded chunks.*
- `Observe`(**cert**) $\rightarrow$ {**true**, **false**} – *Observation is the check for data availability of a blob.* `Observe` *performs a sampling procedure, such as the following: given the blob's certificate, the light node randomly selects a small number of chunks and tries to download them from the network. The output of the observe operation may be a function of the indices for which the light node was able to receive the associated chunk.*

These four operations form the foundation of our model. The `Disperse` and `Retrieve` operations capture the basic functionality of storing and retrieving data, while `Verify` provides cryptographic assurance that data has been properly stored. The `Observe` operation enables lightweight clients to participate in monitoring the system's health without downloading full blobs.

For notational convenience and to avoid ambiguity, we define *blob availability* as follows.

Definition 3.2 (Blob Availability)**.** *A data blob is* available *if any honest user who calls* `Retrieve` *on the blob's certificate during the availability window of the blob can successfully obtain the entire blob.*

Building on the definition of the data availability protocol and blob availability, we now define the key properties that characterize a secure and efficient data availability system:

Definition 3.3 (Safety)**.** *The system has* safety *if every blob for which* `Verify` *returns true can be successfully retrieved during the availability window by any user following the protocol.*

Definition 3.4 (Liveness)**.** *We say that the system is live if every dispersal request submitted to the system eventually receives a valid certificate.*

Definition 3.5 (Observability)**.** *We say that a system is observable with false positive fraction p_{FP} and false negative fraction p_{FN} if, with high probability, the fraction of honest light nodes which observe* `Observe`(**cert**) = **true** *for an unavailable blob is bounded above by p_{FP} and the fraction of honest light nodes which output* `Observe`(**cert**) = **false** *for an available blob is bounded above by p_{FN}.*

These properties provide a framework to evaluate any concrete DA protocol. A good DA protocol should guarantee availability, safety, liveness, and have an observability layer with low false positive and false negative fractions.

3.2 Security Model

In the context of a proof of stake protocol, we can characterize an ODA protocol in terms of various concepts of economic security. Most basically, we can characterize the protocol's tolerance to stake held by a Byzantine adversary. This characterization is based on a model in which each unit of stake is held either by an honest validator that follows the rules of the protocol or by a Byzantine adversary that can take arbitrary adversarial actions. We assume perfect coordination among adversaries and view them as a single entity.

Definition 3.6 (Byzantine Fault Tolerance (BFT) Security Thresholds)**.** *The BFT security thresholds of an ODA are a tuple* (η_S, η_L) *where*

- η_S *(**Safety Threshold**) is the minimum percentage of stake that a Byzantine adversary must control to cause a safety failure, i.e., for a blob to receive a valid certificate without being available, and*
- η_L *(**Liveness Threshold**) is the minimum percentage of stake that a Byzantine adversary must control to cause a liveness failure, i.e., to prevent a request invoked by an honest user from receiving a valid certificate.*

Cryptoeconomic security represents a separate but related notion of economic security which defines an economic slashing penalty faced by any adversary that successfully attacks the system. Cryptoeconomic security can be used to extend the security guarantees of the protocol to the case of rational actors and an adversary which is capable of bribery attacks.

Definition 3.7 (Cryptoeconomic Security)**.** *The cryptoeconomic security of an ODA is defined as the minimum cost that an adversary must incur in order to cause a safety failure to the protocol.*

Establishing cryptoeconomic security for a data availability protocol is made difficult by fundamental game-theoretic issues related to the cost of establishing full consensus about the availability of data, collectively referred to as the "Fisherman's dilemma" [2]. By providing a lightweight `Observe` functionality, an observable data availability protocol reduces the cost for a large community of decentralized light nodes to come to a broad, intersubjective consensus about the availability of a blob. Given the existence of this consensus, this community may initiate a fork of the protocol's staked assets that subjects the adversarial party to a slashing penalty [12], with the result that the protocol obtains cryptoeconomic security in the amount of the slashed funds. We discuss this conversion of observability to cryptoeconomic security in Section 6.2.

4 Protocol Design

EigenDA is an observable data availability service with three layers: storage (validators), access (full nodes), and observability (light nodes), as shown in Section 4.1. Validators provide BFT availability via chunk storage and aggregated attestations; full nodes mediate all inter-layer communication; light nodes observe availability via data availability sampling (DAS). The protocol runs in four phases: dispersal, verification, retrieval, and observation, outlined in Section 4.3. Complementing the protocol, our economic mechanisms (reservations and on-demand fees) fund fixed costs and ensure scalable retrieval (Section 4.4).

4.1 System Architecture

Our protocol operates with several classes of participants, as illustrated in Figure 2, each with specific roles in ensuring data availability:

- **Validators:** Validators provide a Byzantine Fault Tolerant (BFT) guarantee that a blob of data has been stored and will be available to the nodes of the access layer, to be relayed to end users. Functionally, validators store chunks of a blob, validate the correctness of these blobs relative to a commitment, and serve their chunks to full nodes on request. They also return

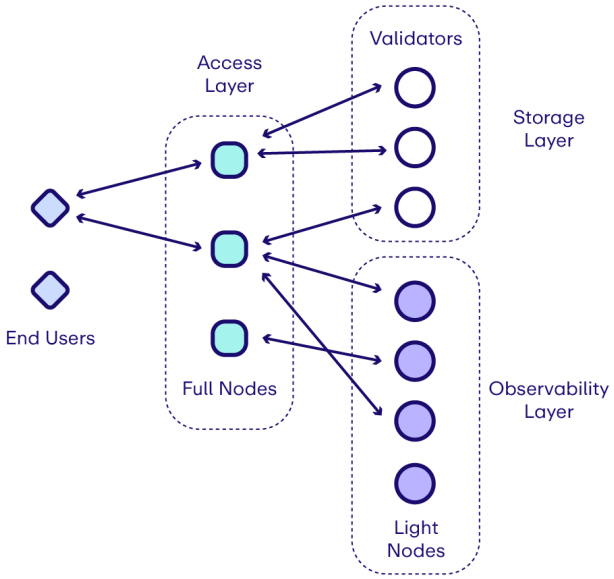

Figure 2: The EigenDA System Architecture. The storage layer, access layer, and observability layer are implemented by validators, full nodes, and light nodes, respectively. All communication between different layers is mediated by the full nodes of the access layer.

a signature which can be aggregated into an attestation of blob availability to be consumed by secure protocols. The BFT availability guarantee of the storage layer makes this layer suitable as a basis of economic security. Stakers can delegate financial stake to validators or collections of validators which they trust. This stake can be slashed via intersubjective forking (see Section 3.2) in the event of a data withholding attack[2].

- **Full Nodes:** Full nodes are specialized nodes introduced in our design to improve availability and scalability. Full nodes play a role in dispersing encoded blobs to the validators, serving reconstructed blobs to end users, and serving collections of individual chunks to light nodes. In normal operation, each blob is primarily stored by at least one full node (the original dispersing full node). Blobs can be replicated to secondary full nodes as well, either proactively or on-demand. Full nodes thus form the backbone of *blob retrievability*, caching data and providing high-throughput read access for users. They are designed to scale out (potentially using cloud infrastructure) to handle large numbers of blob retrievals without bottlenecking the validators and light nodes.
- **Light Nodes:** Light nodes do not store entire blobs, but they play a critical role in observing data availability through sampling. They can be run by any user, stakeholder, or community member who is interested in evaluating the data availability of EigenDA. When a DAS observation is initialized, the light nodes perform the `Observe` protocol for those blobs. Each light node independently selects random chunk indices and tries to download those chunks from any full node. Light nodes only store the small chunks they sampled, and only temporarily

[2]In practice, EigenDA supports the delegation of different types of economic collateral into redundant quorums of validators. In general, some of these quorums may be eligible for forking while others may not.

(e.g., for a couple of hours). Light nodes provide a light-weight means for the community to ascertain the availability of blobs in order to support intersubjective slashing for safety violations.

We design the architecture to scale to a large population of light nodes and, to mitigate spam, may require Sybil-resistant registration (e.g., proof-of-identity).

- **End Users (Clients):** Finally, the end user is anyone who wants to retrieve the blob data (for example, a rollup node that needs the blob for fraud proof generation). The end user does not need to store data long-term; they rely on the above infrastructure to get the blob on demand. Users interact with full nodes for retrieval, and in case of any trouble, they can invoke the sampling mechanism to resolve it (as described in Section 4.3.4).

These components interact in a sequence of phases: **dispersal**, **verification**, **retrieval**, and **observation**, which implement the four functions of the observable data availability protocol in Definition 3.1. We detail each mechanism in Section 4.3.

4.2 Cryptographic and Encoding Primitives

The EigenDA protocol relies on several cryptographic and encoding primitives to ensure data availability with strong security guarantees. We present these primitives in Section A.

The appendix defines a Verifiable Encoding Scheme (VES, See Definition A.2) which is parameterized by a collection of Blob Encoding Parameters.

Definition 4.1 (Blob Encoding Parameters)**.** *Blob encoding parameters are a tuple* (c, γ, r) *where*

- c *(NumChunks) is the total number of encoded chunks after erasure coding (must be a power of 2),*
- γ *(CodingRate) is the ratio of original data to total encoded chunks, providing redundancy (must be an inverse power of 2), and*
- r *(ReconstructionThreshold) is the minimum fraction of total stake required to reconstruct the blob.*

Generally speaking, these parameters determine the trade-off between storage overhead and fault tolerance. A smaller coding rate γ increases redundancy but also storage costs. The reconstruction threshold, r, defines the minimum honest stake needed for data recovery, which must be differentiated from γ to accommodate losses due to the chunk assignment scheme of the encoded chunks. The relationship is given by $r = \frac{c}{c-n}\gamma$, where n is the number of validators; the full derivation is provided in the appendix (Section B).

4.3 Protocol Description

In this part, we provide a detailed description of the dispersal, retrieval, and observation process in our protocol.

4.3.1 Dispersal

Blob dispersal is the process by which a submitted blob is made available across the network. When a new data blob (e.g., a rollup sequencer's block data) is submitted to any full node together with payment (see Section 4.4.1), the protocol proceeds as follows:

1. **Encoding and proving:** The full node will perform the `Encode` and `Prove` operations of the verifiable encoding scheme (Definition A.2).

2. **Metadata distribution:** The full node will distribute the blob metadata (including payment authorization, **commitments**, and a retrieval endpoint) to the validators.

3. **Chunk distribution:** Upon receiving the authorized blob metadata, the validator nodes will calculate their chunk assignment using the `GetAssignments` operation of the chunk assignment scheme (Definition A.5). The validators will then retrieve their assigned chunks from the retrieval endpoint (hosted by the originating full node). Upon receiving and validating their chunks using the `Verify` operation (Definition A.2) of the verifiable encoding scheme, the validators sign a message consisting of the blob metadata to certify their local view of availability using the `SignMessage` operation of the aggregate signature scheme (Definition A.7).

4. **Certificate creation:** The full node constructs a DA certificate by aggregating the validator signatures using the `AggregateSignatures` operation of the aggregate signature scheme.

4.3.2 Verification

Verification is a deterministic check which tells whether a certificate is valid. For EigenDA, verification takes in a security parameters tuple (Definition 3.6), representing the security requirements of the consuming application. An EigenDA certificate is valid if and only if the following conditions are met:

- **Valid cryptographic signature**. The `VerifySignature` call of the aggregate signature scheme must return true, indicating that the signature associated with the certificate is valid.
- **Sufficient signing stake**. The total stake held by the validators represented within the signature must exceed the confirmation threshold, denoted as η_C. The confirmation threshold is set by $\eta_C = (1-\eta_L)$, where η_L is the liveness threshold defined in the security parameters. [3]
- **Valid security parameters**. The blob security parameters and blob encoding parameters must satisfy $1 - \eta_L - \eta_S \geq r$. This ensures the encoding redundancy provided by the blob's encoding settings is sufficient to satisfy the adversarial tolerances specified in the security parameters of the consuming application.

Verification can be performed in whichever context is needed by the consuming application.

4.3.3 Retrieval

Blob retrieval is the process by which end users obtain blob data from the network after it has been dispersed and verified. When a full node disperses a blob, it stores that blob along with the encoded chunks for the duration of the storage period. If a full node receives a retrieval request for a blob or a set of chunks that are not stored locally, it proceeds through a series of mechanisms until it has recovered the requested item.

[3] In line with prior work [15], we separate the adversary threshold into safety and liveness thresholds.

- **Path 1: Pull from full nodes.** The full node sends a query to other full nodes asking if they have the item in question. The node can then send a get request to any full node that is offering to provide the item.

- **Path 2: Recovery from validators.** If no full nodes advertise that they are willing to provide the item in question, the full node can recover it from the validators by downloading the associated chunks and recovering the blob data.

- **Path 3: Recovery from light nodes via DAS activation.** In the unlikely event that Path 2 fails, it indicates that all quorums associated with the item have been corrupted (i.e., malicious validators hold more than a threshold fraction of the stake). At this point, the system must activate the Data Availability Sampling (DAS) protocol to either recover the data or visibly establish the data availability failure to the light nodes in order to support intersubjective slashing.

  DAS is only triggered when a challenge is raised. To activate DAS, an end user who cannot retrieve data can *submit a transaction to the L1* indicating the range of blobs which are unavailable. Once the light nodes have had an opportunity to recover their samples, the full node will attempt to recover the blob by 1) requesting a digest from each light node indicating which samples of which blobs it was able to recover, and 2) constructing and executing a download schedule which observes a per-light node chunk download rate limit. [4]

To summarize the retrieval workflow: the user first tries the simplest route (directly from the disperser). If that fails, the network of full nodes steps in to fetch from validators. If that also fails, the issue is escalated to the observation phase (via the DAS protocol). This tiered approach optimizes for efficiency (most blobs retrieve easily), while still providing a safety net that engages the broader community of light nodes if something seems wrong.

As described in Section 1.3, the full nodes enforce payment and reservation policies and may leverage third-party CDNs (e.g., Cloudflare) to reduce operational cost, increase throughput, and mitigate DDoS attacks. A client may choose a specific full node as a preferred data provider and create a retrieval reservation with this full node. In the event that this full node does not honor a valid retrieval request, the client can choose a different preferred full node as the provider and create a reservation with them. More details on the economic mechanism are provided in Section 4.4.

4.3.4 Observation (via DAS)

The observation process is implemented by the DAS protocol. The DAS protocol is executed by light nodes when activated via a transaction sent to the L1 that identifies a collection of blobs for observation.

For each blob identified by the activation transaction, each light node is assigned a pseudorandom collection of chunk indices based on the light node's ID. The protocol also defines a pseudorandom mapping between blobs and orderings on the set of full nodes. To execute the sampling protocol, a light node will request chunks in turn from each full node in the order defined by this mapping, until it has recovered all of its assigned chunks or until visiting all full nodes. Between

[4] If a light node advertises that it possesses a chunk but then does not provide it, the full node can reconstruct the download schedule and retry. The light node download rate limit is sufficiently over-provisioned that such failures will allow the full node to cull any misbehaving light nodes from a local list before it runs out of download capacity.

requests to full nodes, the light node will wait for a recovery period defined by the full node in order to give the full node time to recover the chunks if it does not already have them. At the end of the DAS protocol, the light node considers a blob available (i.e., `Observe` then returns true) if all sampled chunks are successfully retrieved and validated during the observation time window; otherwise, it deems it unavailable (`Observe` then returns false).

4.4 Economic Mechanisms

EigenDA's approach to economic mechanism design optimizes for providing stable guarantees to both users of the system as well as the validators of the various system components:

- **Guaranteed write throughput for users**: Many protocols have a need to send a steady stream of data to a DA system in order to support the traffic of a live system, such as an L2 rollup. Such protocols should have a means of securing a guarantee of capacity for such stable usage.
- **Guaranteed retrievability for users**: Most protocols that take a dependency on a DA layer rely on a guarantee that data attested as available by the system will be retrievable by some set of watchers that police the activity of the system or relay data to a broader collection of end users. Such watchers should have a means of securing a guarantee of retrieval capacity for all data posted by the protocol that they are servicing.
- **Guaranteed solvency for operators**: Operators of different components of the system will generally need to provision static resources (compute, networking, storage) in order to accommodate any given capacity limit for the system. Many of these resources will incur fixed costs irrespective of usage. In order to facilitate scaling of the network as demand increases, we should endeavor to provide operators with a guarantee that these fixed costs will be covered via fees collected and dispersed by the protocol.

A central design element which links together all of these objectives is that of the reservation. Reservations provide stability for both sides of the reserved relationship: Users can receive guarantees of capacity for both dispersal and retrieval functionalities, while validators receive guarantees of the revenue needed in order to provision infrastructure needed to serve the reserved capacity.

As shown in the introduction, the combination of payment for reading with the specialized networking capacity of the access layer elegantly solves the denial of service problem that haunts most existing DA systems.

4.4.1 Availability Payments

Availability payments are made from users that make data available to validators, full nodes, and light nodes. The payments are meant to cover the fixed global costs associated with making a certain blob available. This includes validator storage, networking, and compute costs, as well as basic keep-the-lights-on costs for full nodes and light nodes.

Availability payments are paid to the EigenDA protocol via an L1 smart contract. Validators receive payments directly from the contract in proportion to their stake. Light nodes, upon uploading a sampled chunk to a full node, receive a random value signed by the full nodes that serves as a "lottery ticket"; a fraction of these tickets are redeemable for rewards on L1, which amortizes the cost of micropayments. Full nodes likewise receive availability payments directly from the contract.

In line with the major economic design principles articulated previously, the primary mechanism for availability payment consists of a long-term reservation that guarantees a certain amount of throughput to the user for the duration of the reservation. If reserved capacity begins to approach the total capacity of the system, this can be used as a signal for the protocol governance to increase capacity by having validators increase their capacities.

For convenience and to support the needs of a greater variety of use cases, EigenDA can also support a flexible mode of on-demand "pay as you go" payments. Such usage has a fixed capacity and is subject to congestion pricing.

4.4.2 Dispersal and Retrieval Payments

Dispersal and retrieval payments are made from users to full nodes through payment channels between them, to cover the costs incurred by full nodes for dispersing blobs and serving them to users, respectively. In each case, these costs may include networking costs as well as computational costs associated with encoding/decoding the blob, or generating cryptographic proofs.

As with availability payments, both reserved and on-demand formats of dispersal and retrieval payments are possible. The purpose of reserved payments in this case is to give end-users additional confidence that their requests will receive preference in the event of congestion, as well as to full nodes the opportunity to scale resources as needed in order to provide guaranteed dispersal and retrieval capacity to end users.

5 Analysis

We now analyze the security and performance characteristics of our scheme, focusing on safety under Byzantine behavior and the false positive and false negative fractions in detecting availability. Our analysis leans on both cryptographic assurance (from the erasure coding and commitments) and statistical assurance (from the sampling procedure), assuming a strong adversary and a large population of honest light nodes.

5.1 Byzantine Security Analysis

The BFT security model ensures system safety and liveness as long as the stake controlled by malicious validators remains below predefined thresholds. We show that our protocol provides strong safety guarantees when using a *sound VES (verifiable encoding scheme)* (Definition A.4) and a *complete CAS (chunk assignment scheme)* (Definition A.6). The sound VES enables each validator to verify the correctness of the chunks they receive from the full node, while the complete CAS scheme guarantees that any subset of validators with more than a percentage r of the stake possess enough chunks to reconstruct the original blob.

The analysis also makes use of the Blob Encoding Parameters (Definition 4.1) and Blob Security Parameters (Definition 3.6). In addition, we refer to a confirmation threshold, η_C, set by $\eta_C = 1 - \eta_L$.

5.1.1 Safety Analysis

In this part, we prove that the safety of our DA protocol is guaranteed when the adversary controls less than $\eta_S = \eta_C - r$ of the total stake.

Theorem 5.1 (Safety Guarantee)**.** *Given a* sound VES*, a* complete CAS*, and at least one honest full node, if the adversary controls less than $\eta_S = \eta_C - r$ percentage of the total stake, where r is the reconstruction threshold of the blob encoding parameters, then whenever a DA certificate is issued, any honest client can retrieve the corresponding blob via full node mediation.*

Proof. Suppose a DA certificate has been issued for a blob. By definition of the confirmation threshold, validators representing at least η_C of the stake signed attestations for this blob. Since the adversary controls at most $\eta_S < \eta_C - r$ of the stake, there exists a set of honest validators H with total stake $\sum_{i \in H} \eta_i \geq \eta_C - \eta_S > r$.

By the *completeness* property of our CAS, for any set of validators H such that $\sum_{i \in H} \eta_i \geq r$, we have $\sum_{i \in H} c_i \geq c\gamma$, where c_i is the number of chunks assigned to validator i, c is the total number of chunks, and γ is the coding rate, such that $c\gamma$ is the number of chunks needed to reconstruct the blob.

Since the validators in H are honest and signed the attestation, they possess all their assigned chunks. The total chunks held by H are sufficient for reconstruction, and by the *soundness* property of our VES, these chunks can be used to reconstruct the original blob. An honest full node can retrieve these chunks from the honest validators in H, reconstruct the blob, and serve it to the requesting client. □

5.1.2 Liveness Analysis

In this part, we prove that the liveness of our DA protocol is guaranteed when the adversary controls less than $\eta_L = 1 - \eta_C$ of the total stake.

Theorem 5.2 (Liveness Guarantee)**.** *Assuming at least one honest full node and that the adversary controls less than $\eta_L = 1 - \eta_C$ of the total stake, any valid blob submission by a client will eventually receive a DA certificate.*

Proof. When a client submits a blob via an honest full node, the honest full node (acting as disperser) encodes and distributes chunks following the protocol. All honest validators will send their signatures upon receiving and verifying their assigned chunks. Since the adversary controls less than $\eta_L = 1 - \eta_C$ of the stake, honest validators control at least η_C of the stake. Therefore, sufficient signatures will be collected to meet the confirmation threshold, and a DA certificate will be issued to the client. □

This completes our BFT security analysis, showing that the combination of a sound VES and complete CAS provides strong safety and liveness guarantees under reasonable adversarial assumptions.

5.2 Observability

We now characterize the detection characteristics (false positive and false negative fractions) of the DAS protocol for EigenDA under a strong adversarial model considered so far, where the adversary observes all sampling requests before choosing which requests to fulfill.

5.2.1 False Positive Fraction Analysis

We consider a powerful adversarial model for the malicious data provider: the adversary first collects the sampling requests from each light node, then, after learning exactly which chunks each node requested, selects a minimal set of chunks that satisfies the requests of as many light nodes as possible.

We formalize the challenge faced by the strong adversary as a game, defined as follows:

Definition 5.3 (Sample Coverage Game)**.** *A sample coverage game with parameters $(m, c, k, \alpha, \gamma)$ is defined as follows: The adversary is given the sampling requests from m light nodes, where each light node samples k random chunks from a total collection of size c. The adversary wins if it can fulfill the sampling requests of at least αm light nodes (tricking them into believing the blob to be available) while revealing fewer than γc unique chunks (so that the blob remains unavailable).*

Each particular instance of the sample coverage game $(m, c, k, \alpha, \gamma)$ corresponds to a sampling pattern $\mathcal{S} = (S_1, \ldots, S_m) \in (\mathcal{C}^k)^m$ where $\mathcal{C}$ is the set of sample indexes with $|\mathcal{C}| = c$. We say that a particular instance of the coverage game is *winnable* if there exists an index set $I \subseteq [m]$ with $|I| \geq \alpha m$ such that $\left|\bigcup_{i \in I} S_i\right| < \gamma c$.

We show that as the total number of light nodes m becomes large, the probability that the sample coverage game is winnable becomes negligible, as formalized in the following theorem.

Theorem 5.4 (Concentration Theorem)**.** *Choose parameters (c, k, α, γ) and $\epsilon > 0$ and let $m \geq \frac{cH(\gamma) - \ln(\epsilon)}{D(\alpha||\gamma^k)}$. Then the sample coverage game with parameters $(m, c, k, \alpha, \gamma)$ is winnable with probability less than ϵ.*

For the full proof of Theorem 5.4 and numerical results, we refer the reader to Section C.2 in the appendix.

We provide a numerical example to explain the meaning of the theorem, with further numerical examples provided in Section C.2.2. For $c = 8192, k = 7, \alpha = 1/8, \gamma = 1/8$, the coverage game is winnable with probability less than $\epsilon = 2^{-128}$ if $m > 2201$. We interpret this result as follows: Suppose we have $m = 2201$ honest light nodes, where each light node samples $k = 7$ random chunks of a blob. There are two possible outcomes:

- Less than $1/8$ of them received all the chunks they request. In this case, $7/8$ of them will consider the data as unavailable, and we assume that social consensus will decide that the data is unavailable.
- At least $1/8$ of them received all the chunks they request. In this case, these $2201 \times 1/8 = 275$ honest light nodes collectively own more than $\gamma c = 1024$ unique chunks with high probability ($> 1 - 2^{-128}$), which means they collectively own enough chunks to recover the blob.

Theorem 5.4 shows that an adversary cannot allow more than a bounded number (αm) of light nodes to output `Observe(cert)` = **true** before the chunks held by these light nodes can be collectively used to reconstruct the blob. However, this result by itself is insufficient to bound the false positive fraction of the system, since we need to ensure that these chunks can be recovered by an honest full node so that the corresponding blob can be passed along to users.

Addressing recoverability requires a more developed version of Theorem 5.4 which ensures that a blob collectively held by the light nodes can be recovered by a full node while considering the upload capacity bounds and associated upload rate limits of light nodes. Specifically, we need a

way for the full node to recover the original blob by requesting a *small* number of chunks from each light node.

Definition 5.5 (Recoverability Game)**.** *A recoverability game with parameters $(m, c, k, \alpha, \gamma)$ is defined as follows: The adversary is given the sampling requests from m light nodes, where each light node samples k random chunks from a total collection of size c. The adversary wins if it can fulfill the sampling requests of at least αm light nodes while preventing a client, who downloads $h = \lceil \frac{\gamma c}{\alpha m} \rceil \leq k$ chunks from each such light node, from being able to collect γc unique chunks and thus recover the blob.*

Each particular instance of the recoverability game, determined by a realized sampling pattern $\mathcal{S} = (S_1, \ldots, S_m) \in (\mathcal{C}^k)^m$, is said to be *recoverable* if the following condition holds: For every index set $I \subseteq [m]$ with $|I| \geq \alpha m$, there exists a download plan $\mathcal{P} = (P_i)_{i \in I} \in (\mathcal{C}^h)^{|I|}$, where each $P_i \subseteq S_i$, such that

$$\left| \bigcup_{i \in I} P_i \right| \geq \gamma c.$$

That is, a client can obtain at least γc unique chunks by downloading h chunks from each of the αm light nodes whose requests were fulfilled. An instance of the recoverability game is said to be *winnable* for the adversary if and only if it is not recoverable.

We prove the following result, corresponding to this definition:

Theorem 5.6 (Recoverability Theorem)**.** *The recoverability game with parameters $(m, c, k, \alpha, \gamma)$ is winnable with probability less than*

$$\sum_{|\mathcal{G}| = \lfloor k/h \rfloor + 1}^{\alpha m} \exp\left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m} || \left(\frac{h|\mathcal{G}|}{c} \right)^k \right) m \right), \tag{1}$$

where $h = \lceil \frac{\gamma c}{\alpha m} \rceil$. For the full proof of Theorem 5.6 and numerical results, we refer the reader to Section C.3 in the appendix.

Theorem 5.6 shows that the EigenDA DAS protocol with parameters (c, k, γ) has a false positive fraction of at most α with bounded probability. We provide a numerical example to illustrate the implications of this theorem, with additional numerical examples provided in Section C.3.2. For $c = 8192, k = 7, \alpha = 1/8, \gamma = 1/8$, the recoverability game is winnable with probability less than $\epsilon = 2^{-128}$ if $m \geq 4096$. This means that, suppose we have 4096 honest light nodes, where each requests 7 random chunks of a blob. There are two possible outcomes:

- Less than 1/8 of them received all the chunks they request. In this case, 7/8 of them will consider the data as unavailable, and we assume that social consensus will decide that the data is unavailable.
- At least 1/8 of them received all the chunks they request. In this case, these $4096 \times 1/8 = 512$ honest light nodes collectively recover the blob by each contributing $h = 2$ unique chunks with high probability ($> 1 - 2^{-128}$).

For the full proof of Theorem 5.6 and numerical results, we refer the reader to Section C.3 in the appendix.

5.2.2 False Negative Fraction Analysis

The false negative fraction of our DAS protocol is low. When the safety threshold of the validator set holds and the blob is available, at least one honest full node can download enough chunks from the validators, reconstruct the original blob, and serve all sample requests from honest light nodes. Consequently, all honest light nodes output $\texttt{Observe}(\textbf{cert}) = \textbf{true}$.

5.2.3 Scalability Analysis

In this section, we first present an analysis of the worst-case communication overhead of the light nodes in the DAS protocol and provide a comparison with the theoretical optimum. Then, we provide an analysis of the worst-case communication overhead of the full nodes.

We reuse the parameters from the previous sections and add a few more parameters:

- n: number of full nodes
- m: number of light nodes
- h: number of samples uploaded to each full node per light node per blob
- t: total system throughput per second
- a: total number of blobs per second
- b: bandwidth per light node
- B: bandwidth per full node

Communication overhead for light nodes We define communication overhead as the total volume of data that must be processed by light nodes divided by the total volume of information made available, denoted as $o = \frac{mb}{t}$.

In our protocol, for each blob, the light node downloads k chunks and then uploads in total $h(n-1)$ chunks to the full nodes. $\frac{t}{a\gamma c}$ is the size of one chunk. Therefore, a light node's bandwidth is given by

$$b = \frac{t}{a\gamma c} \cdot a \cdot (k + h(n-1)) = \frac{t}{\gamma c}(k + h(n-1))$$

Therefore, the overhead is

$$o = \frac{mb}{t} = \frac{m[k + h(n-1)]}{\gamma c}$$

For $k = 8$, $m = 2048$, $h = 2$, $n = 10$, $c = 8192$, $\gamma = 1/8$, we have $o = 52$.

Consider the ideal case where each light node only needs to download exactly h chunks. In this case, the light node's bandwidth is given by

$$b^* = \frac{t}{a\gamma c} \cdot a \cdot (h + h(n-1)) = \frac{t}{\gamma c}(hn)$$

The overhead in the ideal case is

$$o^* = \frac{mb^*}{t} = \frac{mhn}{\gamma c}$$

For $m = 2048$, $h = 2$, $n = 10$, $c = 8192$, $\gamma = 1/8$, we have theoretical optimum $o^* = 40$, which is close to the overhead in our protocol.

Communication overhead for full node We define communication overhead as the total volume of data that must be processed by each full node divided by the total volume of information made available, denoted as $O = \frac{B}{t}$. For each full node, in the worst case, it needs to upload k chunks to m light nodes for each blob. Therefore, the bandwidth is

$$B = \frac{t}{a\gamma c} \cdot a \cdot m \cdot k = \frac{tmk}{\gamma c}$$

The overhead for the full node is

$$O = \frac{B}{t} = \frac{mk}{\gamma c}$$

For $k = 8$, $m = 2048$, $c = 8192$, $\gamma = 1/8$, we have $O = 16$, which is the overhead for the observability layer shown in Figure 1.

6 Discussion

This section discusses key aspects of EigenDA's design. We first compare our DAS adversarial model with other protocols, showing that EigenDA provides security under a stronger adversary that can link requests and adaptively respond. We then examine how EigenDA's observability protocol enables cryptoeconomic security through intersubjective slashing, with full nodes providing evidentiary support for social consensus.

6.1 Adversarial Model Comparison with other DAS protocols

A commonly used assumption in DAS is that each chunk request is unlinkable to its originating light node, which makes the probability of successfully deceiving any individual light node without revealing the blob decay exponentially as the number of requests increases. This amounts to a weak adversary model (called "bulletin board model" [14]), because the adversary is quite restricted by this assumption. However, in practice, this unlinkability assumption is often difficult to guarantee. The "weak model" is thus considered unrealistic at this point, as we mentioned in straw man #4 in Section 1.4.

Another alternative and considerably stronger adversarial model assumes that the adversary can link chunk requests to the originating light nodes. There is existing analysis [1] that derives the minimal number of light nodes required to make it unlikely for an adversary to trick *all* of them into believing the data is available when it is not.

In our analysis, we consider an even stronger adversary. The adversarial data provider (composed of the malicious full nodes) has both the capacity to link together all of the sample requests originating from a given light node and the ability to view the full set of sample requests before choosing how to respond (i.e., which requests to fulfill).

PeerDAS [16] provides a similar analysis as ours, using the same assumptions, in a less formal way. However, as we mentioned earlier, their DAS protocol is based on P2P network, which lacks a formal security guarantee.

6.2 Cryptoeconomic Security

The EigenDA observability protocol provides an upper bound on the percentage of honest light nodes that will falsely believe that an unavailable blob is available. This percentage, denoted $\alpha(m)$ is a decreasing function of the number of honest light nodes, m.

The notion of cryptoeconomic security is generally founded on the assumption that if the percentage of honest light nodes observing a data withholding attack is a sufficiently large supermajority (i.e., $\alpha(m) \ll 1/2$), then the community will perform a successful fork of any intersubjective tokens staked to the validators.

The dependence on honest light nodes reflects the fact that an intersubjective consensus process has access to social/reputational information that may be invoked by members of the community to distinguish between honest and malicious or Sybil light nodes. In theory, a given threshold of light node agreement needed for consensus then translates to a lower bound on the number of honest light nodes that must exist as part of the observer set. In practice, due to the presence of complex social and other informational dynamics which do not make sense to model, we avoid trying to place a lower bound on the actual number of light nodes needed to cryptoeconomically secure the protocol.

It is, however, worth noting that the observability protocol of EigenDA is very well suited to supporting the problem of intersubjective consensus, since the false positive fraction among honest light nodes is itself only a function of the number of *honest* light nodes and is therefore unaffected by malicious Sybils. Thus, to the extent that an intersubjective process is capable of ignoring the input from malicious light nodes, these light nodes have no other capacity to degrade the detection properties of the protocol. This property would generally not be true of observability protocols that make use of a P2P network among light nodes for facilitating the observation of chunk-level availability.

During intersubjective slashing, full nodes may play a critical evidentiary role. They reconstruct blobs by downloading assigned chunks, directly test whether validators serve their chunks, and potentially publish some evidence (for example, logs). Because full nodes are operated by organizations with established operational reputations, their attestations help the community reach social consensus on validator misbehavior. This enables slashing to target faulty validators rather than indiscriminately penalizing all signers.

7 Conclusion

EigenDA shows that separating storage, access, and observability yields a high-throughput, cost-efficient, and cryptoeconomically secure DA system. The storage layer uses a VID scheme with stake to achieve BFT security; the access layer combines economic incentives with elastic infrastructure like CDNs to sustain throughput and resist denial of service; the observability layer provides input for intersubjective slashing that guarantees cryptoeconomic security. Together, these components form a practical DA solution for rollups, AVSs, and other high-throughput applications.

8 Acknowledgment

We would like to express our gratitude to Joachim Neu for reviewing earlier drafts, offering detailed comments, and contributing to discussions that strengthened this work. We also thank Vincenzo Furcillo, Patrick O'Grady, Ertem Nusret Tas, and Lei Yang [5] for their helpful discussions, feedback, and suggestions that improved the design of EigenDA and presentation of this whitepaper.

References

[1] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities. *arXiv preprint arXiv:1809.09044*, 160, 2018.

[2] Vitalik Buterin. A note on data availability and erasure coding. `https://github.com/ethereum/research/wiki/a-note-on-data-availability-and-erasure-coding`, 2025. Accessed: 2025-08-27.

[3] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201. IEEE, 2005.

[4] Cloudflare, Inc. Ddos protection. `https://www.cloudflare.com/ddos/`. Accessed: 2025-10-05.

[5] George Danezis, Giacomo Giuliari, Eleftherios Kokoris Kogias, Markus Legner, Jean-Pierre Smith, Alberto Sonnino, and Karl Wüst. Walrus: An efficient decentralized storage network. *arXiv preprint arXiv:2505.05370*, 2025.

[6] EigenCloud Team. Eigencloud whitepaper. `https://docs.eigencloud.xyz/assets/files/EigenCloud_Whitepaper-127a6743029d2c4858e7633196c47ea4.pdf`, 2025. Accessed: 2025-10-16.

[7] Ethereum Foundation. What is layer 2? `https://ethereum.org/en/layer-2/learn/`, 2025. Accessed: 2025-10-03.

[8] Ethereum.org. Optimistic rollups — data availability. `https://ethereum.org/developers/docs/scaling/optimistic-rollups/#data-availability`. Accessed: 2025-10-05.

[9] Ethereum.org. Zk rollups-data availability. `https://ethereum.org/developers/docs/scaling/zk-rollups/#data-availability`. Accessed: 2025-10-05.

[10] Guy Goren, Andrew Hariri, Timothy DR Hartley, Ravi Kappiyoor, Alexander Spiegelman, and David Zmick. Shelby: Decentralized storage designed to serve. *arXiv preprint arXiv:2506.19233*, 2025.

[11] James Hendricks, Gregory R Ganger, and Michael K Reiter. Verifying distributed erasure-coded data. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 139–146, 2007.

[5] Names are ordered alphabetically by last name

[12] Eigen Labs. EIGEN Token Whitepaper. https://docs.eigencloud.xyz/assets/files/EIGEN_Token_Whitepaper-0df8e17b7efa052fd2a22e1ade9c6f69.pdf, 2024. Accessed: 2025-07-10.

[13] Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 180–197, 2022.

[14] Joachim Neu. Data availability sampling: From basics to open problems. https://www.paradigm.xyz/2022/08/das, 2022. Accessed: 2025-08-27.

[15] Joachim Neu, Srivatsan Sridhar, Lei Yang, and David Tse. Optimal flexible consensus and its application to ethereum. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3885–3903. IEEE, 2024.

[16] Danny Ryan, Dankrad Feist, Francesco D'Amato, Hsiao-Wei Wang, and Alex Stokes. Eip-7594: Peerdas – peer data availability sampling. https://eips.ethereum.org/EIPS/eip-7594, 2024. Accessed: 2025-11-26.

[17] Myna Vajha, Vinayak Ramkumar, Bhagyashree Puranik, Ganesh Kini, Elita Lobo, Birenjith Sasidharan, P Vijay Kumar, Alexandar Barg, Min Ye, Srinivasan Narayanamurthy, et al. Clay codes: Moulding {MDS} codes to yield an {MSR} code. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*, pages 139–154, 2018.

[18] Wikipedia contributors. Hall's marriage theorem. https://en.wikipedia.org/wiki/Hall%27s_marriage_theorem. Accessed: 2025-08-25.

[19] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. {DispersedLedger}:{High-Throughput} byzantine consensus on variable bandwidth networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 493–512, 2022.

Appendices

A Cryptographic and Encoding Primitives

Definition A.1 (Blob Encoding Parameters)**.** *Blob encoding parameters are a tuple* (c, r, γ) *where*

- c *(NumChunks) is the total number of encoded chunks after erasure coding (must be a power of 2),*
- γ *(CodingRate) is the ratio of original data to total encoded chunks, providing redundancy (must be an inverse power of 2), and*
- r *(ReconstructionThreshold) is the minimum fraction of total stake required to reconstruct the blob.*

These parameters determine the trade-off between storage overhead and fault tolerance. A lower coding rate γ increases redundancy but also storage costs, while the reconstruction threshold r defines the minimum honest stake needed for data recovery.

A verifiable encoding scheme uses these parameters to verifiably encode a blob into a set of chunks which can be used to reconstruct the blob in sufficient numbers.

Definition A.2 (Verifiable Encoding Scheme (VES))**.** *A verifiable encoding scheme provides cryptographic primitives for encoding blobs into erasure-coded chunks with verifiable commitments. The scheme consists of six procedures:*

- $\texttt{Encode}(\textbf{blob}, \textbf{params}) \to []\textbf{chunk}$ [6] – *Applies Reed-Solomon erasure coding to extend the blob into a collection of c encoded chunks*
- $\texttt{Recover}([]\textbf{chunk}, \textbf{params}) \to \textbf{blob}$ – *Reconstructs the original blob from a collection of at least* $c\gamma$ *chunks*
- $\texttt{Commit}(\textbf{blob}) \to \textbf{commitments}$ – *Generates cryptographic commitments (e.g., KZG polynomial commitments) to the blob data*
- $\texttt{Prove}(\textbf{blob}, \textbf{params}) \to []\textbf{proof}$ – *Generates opening proofs for each chunk that can be verified against the commitment*
- $\texttt{Verify}(\textbf{commitments}, \textbf{proof}, \textbf{chunk}) \to \{0, 1\}$ – *Verifies that a chunk was correctly derived from the committed blob*
- $\texttt{VerifyLength}(\textbf{commitments}, \textbf{int}) \to \{0, 1\}$ – *Verifies that the commitment corresponds to a blob of the specified length*

Definition A.3 (Correctness of VES)**.** *A VES is correct if the following properties are satisfied:*

1. ***Encoding/Recovery Correctness:*** *For any blob* $\mathbf{b}$ *and encoding parameters* **params***, if* $\textbf{chunks} = \texttt{Encode}(\mathbf{b}, \textbf{params})$ *and* $|\mathbf{S}| \geq c\gamma$ *for any subset* $\mathbf{S} \subseteq \textbf{chunks}$*, then* $\texttt{Recover}(\mathbf{S}, \textbf{params}) = \mathbf{b}$*.*

2. ***Commitment Consistency:*** *For any blob* $\mathbf{b}$ *and encoding parameters* **params***:*

   - *If* $\mathbf{c} = \texttt{Commit}(\mathbf{b})$*,* $\textbf{chunks} = \texttt{Encode}(\mathbf{b}, \textbf{params})$*, and* $\textbf{proofs} = \texttt{Prove}(\mathbf{b}, \textbf{params})$*, then* $\texttt{Verify}(\mathbf{c}, \textbf{proofs}_i, \textbf{chunks}_i) = 1$ *for all valid indices* i*.*

[6] We use []type to indicate a list of values of that type, following the syntax of the Go programming language.

- *If* $\mathbf{c} = \texttt{Commit}(\mathbf{b})$*, then* $\texttt{VerifyLength}(\mathbf{c}, |\mathbf{b}|) = 1$.

Definition A.4 (Soundness of VES)**.** *A VES is sound if it satisfies the following cryptographic properties for any probabilistic polynomial-time (PPT) adversary, except with negligible probability:*

1. ***Polynomial binding.*** *No PPT adversary can produce a commitment* $\mathbf{c}$*, two chunk sets* $\mathbf{S}_0$ *and* $\mathbf{S}_1$ *with* $|\mathbf{S}_0| = |\mathbf{S}_1| \geq c\gamma$ *and associated proofs such that, for both* $j \in \{0, 1\}$*,*

$$\texttt{VerifyLength}(\mathbf{c}, c\gamma) = 1, \texttt{Verify}(\mathbf{c}, \mathbf{proofs}_i^{(j)}, \mathbf{chunks}_i^{(j)}) = 1 \text{ for all } i \in [c\gamma],$$

   while for $\mathbf{b}_j := \texttt{Recover}(\mathbf{S}_j, \mathbf{params})$*,* $\mathbf{b}_0 \neq \mathbf{b}_1$ *or* $\mathbf{b}_0 = \mathbf{b}_1 = \perp$*, except with negligible probability.*

2. ***Length binding.*** *No PPT adversary can produce a commitment* $\mathbf{c}$ *and a blob* $\mathbf{b}$ *with* $|\mathbf{b}| > \ell$ *such that* $\texttt{VerifyLength}(\mathbf{c}, \ell) = 1$ *and* $\texttt{Commit}(\mathbf{b}) = \mathbf{c}$*, except with negligible probability.*

Definition A.5 (Chunk Assignment Scheme (CAS))**.** *A chunk assignment scheme deterministically assigns encoded chunks to validators based on their stake. It provides the following functions:*

- $\texttt{GetAssignments}(\mathbf{params}, []\mathbf{stake}) \to []\mathbf{indices}$ – *Given a blob encoding params and validator stake distribution, returns a mapping from chunk indices to validator indices*

Definition A.6 (Completeness of a CAS)**.** *A CAS is complete if the following properties hold for all possible blob encoding parameters and stake distributions.*

1. ***Unique Assignment:*** *The* $[]\mathbf{indices}$ *output by* `GetAssignments` *must be non-overlapping; no chunk index is assigned to more than one validator.*

2. ***Reconstruction Property:*** *For any set of validators* H *with total stake* $\sum_{i \in H} \eta_i \geq r$*, the chunks assigned to* H *satisfy* $\sum_{i \in H} c_i \geq c\gamma$*, where* c_i *is the number of chunks assigned to validator* i *and* c *and* γ *are as defined in the blob parameters.* r *is the reconstruction threshold that satisfies* $r = \frac{c}{c-n}\gamma$*, where* n *is the number of validators.*

These properties ensure that any coalition of honest validators with sufficient stake (greater than reconstruction threshold) can reconstruct the blob.

We present a concrete chunk assignment scheme and prove its completeness in Appendix B.

Definition A.7 (Aggregate Signature Scheme)**.** *An aggregate signature scheme enables efficient aggregation of validator attestations for data availability certificates. It provides the following operations:*

- $\texttt{SignMessage}(\mathbf{message}, \mathbf{privkey}) \to \mathbf{signature}$ – *Signs a message with a validator's private key to attest to chunk availability*
- $\texttt{AggregateSignatures}([]\mathbf{signatures}) \to (\mathbf{signature}, []\mathbf{nonsigner})$ – *Combines individual signatures into a compact aggregate signature, tracking non-signers*
- $\texttt{VerifySignature}(\mathbf{message}, \mathbf{signature}, []\mathbf{nonsigner}, []\mathbf{signers}) \to \{0, 1\}$ – *Verifies the aggregate signature represents signatures from a list of signers for the given message*

The aggregate signature serves as the cryptographic core of the data availability certificate, providing succinct proof that a quorum of validators has attested to storing their assigned chunks.

B Chunk Assignment Scheme

In this appendix, we present a concrete chunk assignment scheme (CAS) and prove that it satisfies the completeness property.

B.1 Scheme Description

Our chunk assignment scheme (CAS) assigns encoded chunks to validators proportionally to their stake, ensuring that any coalition of validators with sufficient combined stake can reconstruct the blob. Given:

- Blob encoding parameters (c, γ, r) where c is the total number of chunks, γ is the coding rate, and r is the reconstruction threshold
- A set of n validators with stakes $\eta_1, \eta_2, \ldots, \eta_n$, where $\sum_{i=1}^{n} \eta_i = 1$

The assignment algorithm `GetAssignments` works as follows:

1. **Initial allocation:** For each validator i, calculate the base number of chunks:
$$c'_i = \lceil \eta_i (c - n) \rceil$$
2. **Total initial assignment:** Compute $c' = \sum_{i=1}^{n} c'_i$
3. **Sort validators:** Order validators deterministically and assign index k_i to each validator i
4. **Final assignment:** The number of chunks assigned to validator i is:
$$c_i = c'_i + \mathbb{I}_{k_i \leq c - c'}$$
where $\mathbb{I}$ is the indicator function that adds one extra chunk to the first $c - c'$ validators to ensure the total number of assigned chunks equals c.

B.2 Completeness Proof

We prove that the scheme described above has the completeness property.

Theorem B.1 (Assignment Scheme Completeness)**.** *The chunk assignment scheme described above is complete according to Definition A.6.*

Proof. We verify both required properties:

1. Unique Assignment: By construction, each chunk is assigned to exactly one validator. The algorithm assigns c_i chunks to validator i, and we have:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} (c'_i + \mathbb{I}_{k_i \leq c - c'}) = c' + (c - c') = c$$

Since we assign exactly c chunks in total and each validator receives a distinct set of c_i chunks, no chunk index is assigned to more than one validator.

2. Reconstruction Property: We need to show that for any set of validators H with total stake $\sum_{i\in H}\eta_i \geq r = \frac{c}{c-n}\gamma$, the chunks assigned to H satisfy $\sum_{i\in H} c_i \geq c\gamma$.

First, we establish a lower bound on c_i for each validator i. By construction:

$$c_i \geq c_i' = \lceil \eta_i(c-n) \rceil \tag{2}$$
$$\geq \eta_i(c-n) \tag{3}$$

Then, for any set H with $\sum_{i\in H}\eta_i \geq r = \frac{c}{c-n}\gamma$:

$$\sum_{i\in H} c_i \geq \sum_{i\in H}\eta_i(c-n) \tag{4}$$
$$= (c-n)\sum_{i\in H}\eta_i \tag{5}$$
$$\geq (c-n)\cdot\frac{c}{c-n}\gamma \tag{6}$$
$$= c\gamma \tag{7}$$

Thus, any coalition of validators with combined stake at least r has sufficient chunks to reconstruct the blob. □

C DAS Analysis

In this part, we provide a summary of the notation and proof of Theorem 5.4 and Theorem 5.6.

C.1 Summary of Notations

In this section, we summarize the setup for Theorem 5.4 and Theorem 5.6.

Recall that during dispersal, the blob will be encoded using the Verifiable Encoding Scheme into a collection of $\mathcal{C}$ chunks, with $|\mathcal{C}| = c$. The key property of the encoding is that for any collection $\mathcal{U} \subset \mathcal{C}$ with $|\mathcal{U}| \geq \gamma c$, we can use the chunks in this collection to recover the original blob.

When the observability protocol is activated for a given blob, each honest light node will attempt to sample k random chunks of that blob from the full nodes. We let $\mathcal{M}$ represent the set of honest light nodes, with $|\mathcal{M}| = m$. We let $\mathcal{S} = (S_1, \ldots, S_m) \in (\mathcal{C}^k)^m$ denote the assignment of light nodes to sample indices for a particular blob, with $S_i \subset \mathcal{C}^k$ denoting the set of indices assigned to light node i.

A light node will return `Observe` = **true** if and only if they are able to download each chunk in S_i within a designated period, and will otherwise return **false**. We let $\mathcal{N} \subseteq \mathcal{M}$ denote the set of light nodes that return **true**, and assume that the adversary has full control over $\mathcal{N}$ by selectively serving the light nodes.

For each blob, a light node will share any of its available chunks with each full node up to h times. [7]

The following table summarizes the parameters appearing in the above setup:

[7] In practice, this is implemented by a rate limit which limits the rate at which each full node can download chunks from light nodes.

Symbol	Description	Example Value
c	The total number of chunks after encoding ($c = \|\mathcal{C}\|$)	8192
γ	Coding rate. Fraction of chunks needed to recover the original blob. That is, blob can be recovered from γc chunks.	1/8
m	The total number of honest light nodes ($m = \|\mathcal{M}\|$)	≥ 4096
k	Number of chunks sampled per blob by each light node ($k = \|\mathcal{S}_i\|$)	7
h	Number of chunk uploads per blob per full node by each light node	2

C.2 Concentration Theorem

In this part, we provide the proof of the concentration theorem provided in Section 5.2 and the related numerical results.

C.2.1 Proof of the Concentration Theorem

We begin with a technical lemma that characterizes the concentration properties of random samples drawn from a finite set of chunks. Specifically, we show that for any subset of light nodes $\mathcal{N} \subset \mathcal{M}$ with size $\geq a$, the probability that the number of unique chunks sampled by all of the light nodes in $\mathcal{N}$ is less than b is bounded.

Lemma C.1 (Concentration Lemma)**.** *Let $\mathcal{S}$ be drawn uniformly at random from $(\mathcal{C}^k)^m$. Then the following bound holds as long as $\frac{a}{m} > \left(\frac{b}{c}\right)^k$:*

$$\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq a \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq b\right) \leq \exp\left(cH\left(\frac{b}{c}\right) - D\left(\frac{a}{m} || \left(\frac{b}{c}\right)^k\right) m\right) \tag{8}$$

Proof. We will proceed in three steps. First, we consider an upper bound on the number of possible ways to choose b chunks. Second, for each specific set of b chunks, we show that the probability of finding a samples whose union falls in the set is bounded. Finally, we combine the previous two steps to argue that, the probability that there exist a set of a samples whose union is of size less than or equal to b is bounded.

Step 1: Let C_0 be a set of b chunks. There are $\binom{c}{b}$ different ways of choosing C_0. Utilizing the information-theoretic bound, we have

$$\binom{c}{b} \leq e^{cH(\frac{b}{c})} = \exp(cH(\frac{b}{c}))$$

where $H(\frac{b}{c}) = -[\frac{b}{c}\ln(\frac{b}{c}) + (1-\frac{b}{c})\ln(1-\frac{b}{c})]$.

Step 2: Let $\mathcal{X}$ be the set of light nodes such that all k of their samples fall completely within the set C_0. Note that $|\mathcal{X}| \geq a \Leftrightarrow \exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq a \wedge \cup_{i\in\mathcal{N}} \mathcal{S}_i \subseteq C_0$. We aim to derive the probability that $|\mathcal{X}| \geq a$. Since each light node is sampling k randomly chosen chunks, the probability that one of its samples falls into C_0 is $\frac{b}{c}$. Since each sample is independently chosen, the probability that all the k chunks sampled by a given light node is fully contained within C_0 is $p = (\frac{b}{c})^k$. Furthermore, these events are independent among the light nodes. Therefore, $|\mathcal{X}|$, the number of

light nodes whose samples fall completely within C_0, is a binomial random variable (i.e., the sum of m independent Bernoulli random variables with parameter p).

We can make use of the Chernoff-Hoeffding bound for $|\mathcal{X}|$. By substituting $p+\epsilon$ with $\frac{a}{m}$ in the additive form of the Chernoff–Hoeffding bound, we obtain the inequality:

$$\Pr(|\mathcal{X}| > a) \leq \exp\left(-D\left(\frac{a}{m}\middle\|p\right)m\right),$$

where $\frac{a}{m} > p$, as required for the bound to apply. Here, $D(\frac{a}{m}\|p)$ denotes the Kullback–Leibler (KL) divergence between Bernoulli distributions with parameters $\frac{a}{m}$ and p, given explicitly by $D\left(\frac{a}{m}\middle\|p\right) = \frac{a}{m}\ln\left(\frac{\frac{a}{m}}{p}\right) + \left(1-\frac{a}{m}\right)\ln\left(\frac{1-\frac{a}{m}}{1-p}\right)$.

Recall that $p = (\frac{b}{c})^k$, and $|\mathcal{X}| \geq a \Leftrightarrow \exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq a \wedge \bigcup_{i\in\mathcal{N}} \mathcal{S}_i \subseteq C_0$ which gives:

$$\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq a \wedge \bigcup_{i\in\mathcal{N}} \mathcal{S}_i \subseteq C_0\right) \leq \exp\left(-D\left(\frac{a}{m}||\left(\frac{b}{c}\right)^k\right)m\right),$$

where $\frac{a}{m} > \left(\frac{b}{c}\right)^k$.

Step 3: Thus, by the union bound, the total probability that there exist $\alpha_1 m$ samples whose union is of size less than or equal to γc is bound by:

$$\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq a \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq b\right) \tag{9}$$

$$\leq \exp\left(cH\left(\frac{b}{c}\right)\right) \cdot \exp\left(-D\left(\frac{a}{m}||\left(\frac{b}{c}\right)^k\right)m\right) \tag{10}$$

$$= \exp\left(cH\left(\frac{b}{c}\right) - D\left(\frac{a}{m}||\left(\frac{b}{c}\right)^k\right)m\right) \tag{11}$$

□

Next, we provide proof of the concentration theorem we proposed in Section 5.2.

Theorem C.2 (Sample Coverage)**.** *Choose parameters* (c, k, α_1, γ) [8] *and* $\epsilon > 0$ *and let* $m \geq \frac{cH(\gamma)-\ln(\epsilon)}{D(\alpha_1||\gamma^k)}$ *and* $\alpha_1 > \gamma^k$*. Then the sample coverage game with parameters* $(m, c, k, \alpha_1, \gamma)$ *is winnable with probability less than* ϵ*.*

Proof. Suppose there's a set of honest light nodes with size m. We want to make sure that the probability that a portion of honest light nodes, denoted as $\alpha_1 m$, can be convinced that the data is available, while the blob is not collectively available to them, is negligible. In other words, we would like to make sure that the probability that there exists a set of more than $\alpha_1 m$ honest light nodes whose samples are contained within a set of size $\gamma c - 1$ is smaller than a negligible number ϵ. That is:

$$\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq \alpha_1 m \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq \gamma c - 1\right) \leq \epsilon$$

[8] In the main text (Definition 5.3), we use α to denote the false positive fraction parameter. In the appendix, we use α_1 for the sample coverage theorem and α_2 for the load-balanced recovery theorem to distinguish between the two different (though related) analyses. Both represent the same conceptual parameter: the maximum fraction of honest light nodes that may falsely observe availability.

Given the condition of the theorem

$$m \geq \frac{cH(\gamma) - \ln(\epsilon)}{D(\alpha_1||\gamma^k)},$$

we derive

$$\exp(H(\gamma)c - D(\alpha_1||\gamma^k)m) \leq \epsilon.$$

On the other hand,

$$\begin{aligned} &\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq \alpha_1 m \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq \gamma c - 1\right) \\ &\leq \Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq \alpha_1 m \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq \gamma c\right) \\ &\leq \exp(H(\gamma)c - D(\alpha_1||\gamma^k)m) \end{aligned} \tag{12}$$

We use Lemma C.1 with the substitutions $a = \alpha_1 m$ and $b = \gamma c$ in the second step of Equation (12). Note that we require $\alpha_1 > \gamma^k$ for Lemma C.1 to apply. Therefore, we conclude

$$\Pr\left(\exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq \alpha_1 m \wedge \left|\bigcup_{i\in\mathcal{N}} \mathcal{S}_i\right| \leq \gamma c - 1\right) \leq \epsilon,$$

which means that the probability of winning the sample coverage game with parameters $(m, c, k, \alpha_1, \gamma)$ is less than ϵ. □

C.2.2 Numerical Results

By evaluating the condition in Theorem C.2 at the parameters used in our protocol, we derive the numerical result below.

In EigenDA, $c = 8192$. We choose $\epsilon = 2^{-128}$ and get $-\ln(\epsilon) = 88.73$. m should satisfy:

$$m \geq \frac{cH(\gamma) + 88.73}{D(\alpha_1||\gamma^k)}$$

We derived the numerical bound for m, given that $\gamma = 1/8$ and k, α_1 specified in the table below:

k	α_1	m
4	1/8	4789
5	1/8	3441
6	1/8	2685
7	1/8	2201
8	1/8	1865
12	1/8	1158
16	1/8	840
20	1/8	659
24	1/8	542
8	1/8	1865
8	1/6	1368
8	1/4	883
8	1/3	647

The code for calculating the results shown in the table above can be found here.

C.3 Recoverability Theorem

In this part, we provide the proof of the recoverability theorem provided in Section 5.2 and the related numerical results.

C.3.1 Proof of the Recoverability Theorem

The recoverability theorem shows that an adversary is unable to prevent a blob from being recovered when recovery efforts are subject to rate limits imposed by light nodes.

Recall that, given the protocol parameters $(m, c, k, \alpha, \gamma)$, a sampling pattern $\mathcal{S} = (S_1, \ldots, S_m) \in (\mathcal{C}^k)^m$, is said to be *recoverable* if the following condition holds: For every index set $I \subseteq \mathcal{M}$ with $|I| \geq \alpha m$, there exists a download plan $\mathcal{P} = (P_i)_{i \in I} \in (\mathcal{C}^h)^I$, where each $P_i \subseteq S_i$, such that

$$\left| \bigcup_{i \in I} P_i \right| \geq \gamma c.$$

This ensures that a client can obtain at least γc unique chunks by downloading h chunks from each of the αm light nodes whose requests were served by the adversary.

A relevant concept from the field of combinatorics is the idea of a generalized transversal. A generalized transversal with parameter h for a collection $\mathcal{S} = (\mathcal{S}_i)_{i \in I}$ (given some index set $I \subseteq \mathcal{M}$) is a set $\mathcal{X}$ which contains exactly h unique elements from each set in $\mathcal{S}$ (that is, by choosing $\mathcal{P}_i \subset \mathcal{S}_i$ such that $|P_i| = h$ and $|\cup_{i \in I} \mathcal{P}_i| = h|I|$). Notice that for a collection of light nodes $I \subseteq \mathcal{M}$ with $|I| = \alpha m$, if $\mathcal{S} = (\mathcal{S}_i)_{i \in I}$ has a generalized transversal with parameter $h \geq c\gamma/\alpha m$, then $|\cup_{i \in I} \mathcal{P}_i| = h|I| \geq c\gamma$, and it follows that a client able to download h samples from each light node in the set will be able to collect $h\alpha m \geq c\gamma$ unique chunks and thus reconstruct a blob.

Thus, to show that the recoverability condition holds in general, it suffices to show that with high probability, every collection $(\mathcal{S}_i)_{i \in I}$ with $|I| = \alpha m$ has a generalized transversal with parameter $h = \left\lceil \frac{\gamma c}{\alpha m} \right\rceil$. (Note that recoverability for a set of light nodes I with $|I| > \alpha m$ that have received their chunks follows trivially, e.g., by taking any subset of size αm).

We now show that the probability of the adversary winning the data recovery game is upper bounded by establishing an upper bound on the probability that there exists a collection $(\mathcal{S}_i)_{i \in I}$ with $|I| = \alpha m$ that does not admit a generalized transversal with parameter h.

Theorem C.3 (Load-balanced Recovery)**.** *The recoverability game with parameters $(m, c, k, \alpha, \gamma)$ is winnable with probability less than*

$$\sum_{|\mathcal{G}| = \lfloor k/h \rfloor + 1}^{\alpha_2 m} \exp\left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m} || \left(\frac{h|\mathcal{G}|}{c} \right)^k \right) m \right), \tag{13}$$

where $h = \left\lceil \frac{\gamma c}{\alpha m} \right\rceil$, given that $\frac{|\mathcal{G}|}{m} > \left(\frac{h|\mathcal{G}|}{c} \right)^k$.

Proof. We can derive the upper bound on the probability that there exists a collection $(\mathcal{S}_i)_{i \in I}$ with $|I| = \alpha m$ that does not admit a generalized transversal with parameter h using the generalized form of Hall's Theorem. The theorem states that a generalized transversal exists when the generalized Hall's marriage condition holds (see its proof in Section C.3.3).

The generalized Hall's marriage condition holds if for each collection of sets $\mathcal{G} \subseteq \mathcal{S}$, the union of these sets is of greater size than the number of sets times the number of chunks provided by each node, i.e., $|\bigcup_{S\in\mathcal{G}} S| \geq h|\mathcal{G}|$. In our case, it is sufficient to show that this condition holds for $|\mathcal{G}| = 1, \ldots, \alpha_2 m$, since $|\mathcal{G}| \leq |\mathcal{S}| = |I| = \alpha_2 m$. Based on the size of $|\mathcal{G}|$, we consider the following two cases:

1. For $|\mathcal{G}| = 1, \ldots, \lfloor k/h \rfloor$, $h|\mathcal{G}| \leq k$. Since $|S| = k$ for any $S \in \mathcal{G}$, $|\bigcup_{S\in\mathcal{G}} S| \geq \min(|S_i|) = k$. Therefore, $|\bigcup_{S\in\mathcal{G}} S| \geq h|\mathcal{G}|$ holds with probability 1.
2. For $|\mathcal{G}| = \lfloor k/h \rfloor + 1, \ldots, \alpha_2 m$, we will prove the probability that $\exists \mathcal{G}, |\bigcup_{S\in\mathcal{G}} S| < h|\mathcal{G}|$ is negligible.

We observe that the condition under which the generalized Hall's theorem fails to hold can be restated in the form of the condition in Lemma C.1:

$$\exists \mathcal{G}, |\bigcup_{S\in\mathcal{G}} S| < h|\mathcal{G}| \Leftrightarrow \exists \mathcal{N} \subset \mathcal{M} : |\mathcal{N}| \geq |\mathcal{G}| \wedge |\cup_{i\in\mathcal{N}} \mathcal{S}_i| < h|\mathcal{G}|,$$

where $|\mathcal{G}| = \lfloor k/h \rfloor + 1, \ldots, \alpha_2 m$.

By applying Lemma C.1 with the substitutions $a = |\mathcal{G}|$ and $b = h|\mathcal{G}|$, we obtain:

$$\Pr(\exists \mathcal{G}, h|\mathcal{G}| > |\bigcup_{S\in\mathcal{G}} S|) \leq \sum_{|\mathcal{G}|=\lfloor k/h \rfloor+1}^{\alpha_2 m} \exp\left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m}||\left(\frac{h|\mathcal{G}|}{c}\right)^k\right) m\right) \tag{14}$$

Therefore, according to the generalized Hall's theorem, the probability that some subset of samples fails to admit a generalized transversal with parameter h is bounded by the inequality above. As we discussed earlier, the existence of such transversal implies that the data is recoverable. Therefore, the adversary can win the recoverability game with probability less than

$$\sum_{|\mathcal{G}|=\lfloor k/h \rfloor+1}^{\alpha m} \exp\left(cH\left(\frac{h|\mathcal{G}|}{c}\right) - D\left(\frac{|\mathcal{G}|}{m}\middle\|\left(\frac{h|\mathcal{G}|}{c}\right)^k\right) m\right)$$

□

C.3.2 Numerical Results

We choose the negligible number $\epsilon = 2^{-128}$. Assuming that $m \leq c$ [9], the number of terms in the summation above is at most $\alpha_2 c$. Assuming that $c = 8192, \alpha_2 \leq 1/2$, we have $\alpha_2 m \leq 4096 = 2^{12}$. Using Equation (13), a sufficient condition for the adversary to win the recovery game with probability less than ϵ is:

$$\max_{|\mathcal{G}|=\lfloor k/h \rfloor+1,\cdots,\alpha_2 m} \exp\left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m}||\left(\frac{h|\mathcal{G}|}{c}\right)^k\right) m\right) \cdot 2^{12} \leq 2^{-128} \tag{15}$$

$$\Rightarrow \max_{|\mathcal{G}|=\lfloor k/h \rfloor+1,\cdots,\alpha_2 m} \exp\left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m}||\left(\frac{h|\mathcal{G}|}{c}\right)^k\right) m\right) \leq 2^{-140} \tag{16}$$

$$\Rightarrow \max_{|\mathcal{G}|=\lfloor k/h \rfloor+1,\cdots,\alpha_2 m} \left(cH(\frac{h|\mathcal{G}|}{c}) - D\left(\frac{|\mathcal{G}|}{m}||\left(\frac{h|\mathcal{G}|}{c}\right)^k\right) m\right) \leq -98 \tag{17}$$

[9] We will check that this actually holds when getting the results

We used the fact that $\exp(-98) < 2^{-140}$ in the reasoning above.

For the k specified in the table below, we numerically compute the minimum m and show them in the table below:

k	α_2	m	h
5	1/8	8192	1
6	1/8	8192	1
7	1/8	4096	2
8	1/8	4096	2
12	1/8	2048	4
16	1/8	1171	7
20	1/8	820	10
24	1/8	683	12
8	1/8	4096	2
8	1/6	3072	2
8	1/4	2048	2
8	1/3	1536	2

The code for calculating the results shown in the table above can be found here.

C.3.3 Proof of Generalized Hall's Theorem

There are two equivalent formulations of Hall's theorem: the combinatorial formulation and the graph theoretic formulation [18]. In our analysis, we used the combinatorial formulation of the generalized Hall's theorem. We'll prove the generalized Hall's theorem based on Hall's theorem using the graph theoretic formulation.

Theorem C.4 (Hall's Theorem, Graph Theoretic Formulation)**.** *A bipartite graph G with bipartition $\{A, B\}$ contains a matching of A if and only if $|N(S)| \geq |S|$ for all $S \subseteq A$.*

Theorem C.5 (Generalized Hall's Theorem, Graph Theoretic Formulation)**.** *A bipartite graph G with bipartition $\{A, B\}$ contains a k-matching of A if and only if $|N(S)| \geq k|S|$ for all $S \subseteq A$.*

Proof. Split each node $a \in A$ into k distinct nodes: $a_1, a_2, \ldots, a_k$. Let the set of all these split nodes form a new set A'. Each node $a_i \in A'$ is connected to all the original neighbors of a in G. In this way, we construct a new graph G' with bipartition $\{A', B\}$.

Now, consider any subset $S' \subseteq A'$. Let $S \subseteq A$ be the set of original nodes corresponding to the preimages of the nodes in S'. By construction, the neighbors of S' in G' are also neighbors of S in G. Also, since each node $a \in A$ contributes at most k split nodes to A', we have: $|S| \geq \lceil |S'|/k \rceil$. Using the assumption that $|N(S)| \geq k|S|$, we get: $|N(S')| = |N(S)| \geq k|S| \geq k\lceil |S'|/k \rceil \geq |S'|$

Thus, the condition of Hall's theorem is satisfied for the graph G' with bipartition $\{A', B\}$. Therefore, by Hall's theorem, G' contains a matching that covers all nodes in A'.

Since each node in A was split into k nodes in A', this matching corresponds to each node in A being matched to k distinct neighbors in B in the original graph G. Hence, G contains a k-matching of A, which proves the generalized Hall's theorem. □