BINANCE RESEARCH

Why You Should Care About Data Availability

April 2024

Derek Ho and
Shivam Sharma

Table of Contents

1 Key Takeaways

- The data availability (“DA”) problem asks this question: how can we ensure that all network participants can access the data of a newly proposed block? It emerged as a necessity alongside rollups and the modular blockchain paradigm.

- Dedicated DA layers have largely emerged in response to the increasing costs of posting data directly to major Layer-1s (“L1s”), especially Ethereum.

- This report is part of our **technical series**. We feature major DA players and explain the technical primitives underpinning their solutions and their anticipated impacts on products and future positioning.

- **Celestia** and **Avail** (and likely **Danksharding**) rely on **data availability sampling (“DAS”)**. With DAS, light clients contribute to security very efficiently. A dozen or so DAS operations give over 99.9% confidence that all data is available. This results in **very high security**, where data availability is guaranteed as long as 1-of-n or even 0-of-n full nodes are honest.

- Among the protocols discussed in this report, **EigenDA** is unique in its use of a **dispersal protocol**. This may enable **better scalability** as individual nodes’ storage requirements drop as more nodes join the network. The downside is that security requires a majority or supermajority of honest nodes.

- **Avail and EiganDA** (and Danksharding) use Kate-Zaverucha-Goldberg **(“KZG”)**, which has useful properties that allow validity proofs to be generated. Avail uses this for **faster DA finality** and potentially better integration with zk-rollups.

- All the above projects use **erasure codes** extensively. This technique allows protocols to achieve better (often optimal) trade-offs.

- In designing these protocols, tradeoffs include **scalability**, **security**, and **finality speed**.

- The different protocols featured in this report made different design decisions regarding these trade-offs. This will likely result in persistent differences in cost, security, and finality speed, which may be deciding factors for project teams (although KPI comparisons will likely fluctuate in the short term).

2 Introduction

The era of modular blockchains is upon us and remains one of the overarching themes of the crypto zeitgeist of the early 2020s. While Layer-2 ("L2") rollups, whether optimistic, zero knowledge ("zk"), or other, have seen their fair share of headlines, one less discussed part of the puzzle is the data availability ("DA") layer.

DA is primarily a developer-oriented concept; the user of a DA layer is a rollup developer, not the end user. Users do not typically interact with transaction data, and in many cases, some might not feel particularly bothered about where and how their data is published or made available (look no further than many contemporary Web2 applications with billions of users). However, the impact of DA is huge, as it is the key determinant of a rollup's costs, which undoubtedly affects the end user. Lower DA costs translate directly into lower rollup costs, not only improving the user experience but also opening up new areas of innovation for developers.

The Ethereum rollup DA market is currently dominated by native Ethereum DA. This dominance is expected, given that the only other major alternative, Celestia, went live fairly recently. Other than Celestia, we also have the likes of EigenDA and Avail, alongside some others, on the horizon. On the Layer-1 ("L1") side, Ethereum recently completed its Dencun upgrade, which featured EIP-4844 (also known as "Proto-Danksharding"), a long-awaited upgrade targeted specifically at reducing Ethereum's native DA costs. The impact of that is recently being made clear in reduced L2 rollup transaction fees. A key theme to watch for the future will be whether dedicated DA layers like Celestia will win out or eventually be dethroned by Ethereum's long-term vision of full Danksharding.

In this report, we start by examining the basics of data availability and why it matters. We then cover Ethereum's recent Dencun upgrade before diving into the dedicated DA layers (Avail, Celestia, and EigenDA). We provide a technical overview of the underlying primitives used by each project and provide some comparative analysis. We end the report with an outlook for the DA market and some interesting questions to consider as things evolve.

This is part of Binance Research's Technical Series. The initial sections lay out the landscape and key players in this space. We gradually focus more on technical topics, ultimately diving into the key primitives underpinning the main players' solutions. Those with a technical background may find these sections more to their interest.

Note: We may use the term "rollup" as a catch-all to refer to various types of L2 scaling solutions, including rollups, validiums, and optimiums. More details about the exact differences between these solutions can be found on the L2 Beat and Ethereum websites.

3 Modular Thesis Refresher

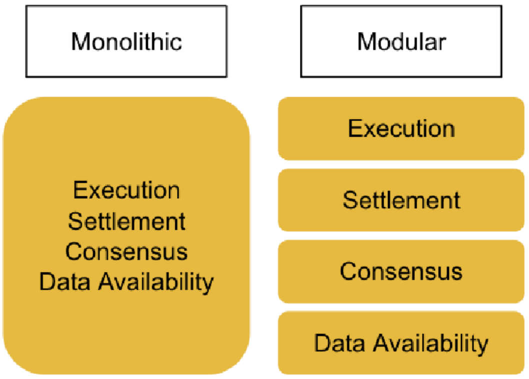
Setting the Scene: Monolithic vs. Modular

Before diving into the modular blockchain thesis, let's quickly recap the concept of monolithic blockchains. Defining a blockchain at its most basic level as an **immutable ledger of transactions**, we can broadly classify the majority of blockchains, at least those with notable value attached to them, as monolithic blockchains. To meet its **fundamental purpose of recording valid transactions and data chronologically**, a blockchain must perform four critical functions:

- **Execution:** processing transactions to update the state of the blockchain.
- **Settlement:** resolving disputes, verifying the validity of transactions, and ensuring the finality of transactions.
- **Consensus:** reaching an agreement between validators or miners on transaction ordering, e.g., Proof-of-Stake ("PoS"), Proof-of-Work ("PoW"), etc.
- **Data Availability ("DA"):** ensuring transaction data is available for the entire network to view.

Monolithic blockchains, such as Ethereum and Solana, perform all of these functions on the same layer and in a generalized manner. **Modular blockchains, on the other hand, seek to separate these functions across multiple different chains**.

Figure 1: Monolithic vs. modular blockchains



Source: Binance Research

Why Modular?

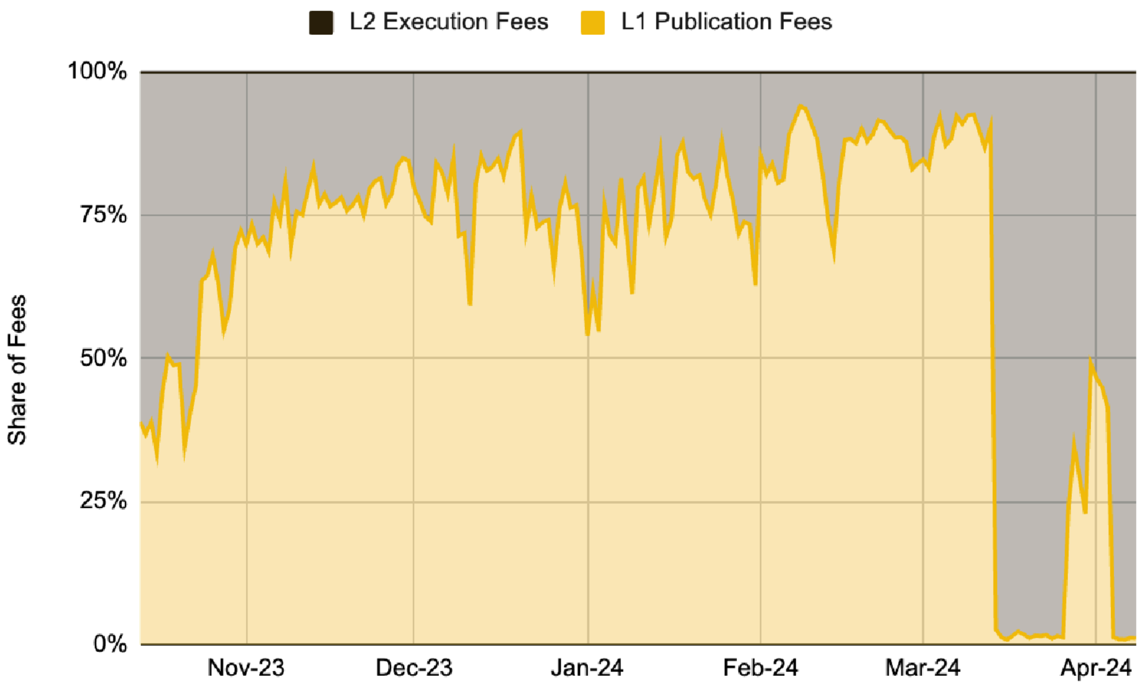
The crux of the modular blockchain thesis is about the efficiency generated through the **separation of roles**. Instead of a single network handling all core functions required for a functioning blockchain, the **modular approach advocates for a separation of duties**, such as execution or DA, across specialized networks.

By separating the different components of a monolithic L1, a modular blockchain can be optimized for specific functions at each layer of the stack, thereby enhancing decentralization, security, and scalability where necessary. The rationale is that the sum of these layers will be able to achieve higher levels of customization and efficiency.

Initially gaining traction with **Ethereum's shift towards a rollup-centric architecture,** the modular approach has become an increasingly visible part of the crypto ecosystem over the last few years. Today, Layer-2 ("L2") rollups, which enable the separation and optimization of the execution environment, represent the most popular type of modular blockchain. L2 rollups like Arbitrum One and OP Mainnet execute and bundle (or 'roll up') numerous user transactions into a single transaction, which is then submitted to the L1 (Ethereum in this case) for settlement.

While L2s have been growing with strength, one of the **critical bottlenecks in realizing their full scalability has been in the DA layer**. L2 rollups post their data to Ethereum using 'calldata', which is neither **optimized for L2s nor scalable to meet their DA needs**. (The DA problem is explained in more detail below.) In fact, as depicted in Figure 2 below, historically, **upwards of 70-90% of rollup transaction fees can be attributed to these data posting costs**.

Figure 2: L1 publication fees have historically been the dominant cost driver for rollups, particularly optimistic rollups



Source: Source: Dune Analytics (@optimismfnd), Binance Research, as of April 9, 2024

Thus, there has been a **strong focus on advancing DA capabilities**, as evidenced by the **emergence of dedicated DA solutions** such as Celestia, EigenDA, and Avail, among others. Ethereum's recent Dencun upgrade, which included EIP-4844, was also primarily focused on improving Ethereum's DA capabilities. Before we dive into these projects and EIP-4844 further, let's explore a brief technical overview of DA.

Data Availability Explained

In any blockchain system, the **DA layer is responsible for ensuring that transaction data is systematically made available on-chain**. This is critical for maintaining the chain's liveness and integrity and verifying the validity of transactions.

In practice, every monolithic L1 blockchain implicitly ensures DA. For example, all Ethereum full nodes download all transaction data in the process of validating a new block, which itself ensures DA for all of the transactions. This is somewhat of a background process that has historically been given relatively little attention. However, the proliferation of L2 rollups and the modular thesis have given the issue more visibility.

Specifically, rollups highlight the DA issue because of how they operate. As a reminder, **rollups execute transactions away from the L1 and then post their transaction data to it.** There are two types of rollup solutions: **optimistic and zero knowledge ("zk")**. At a high level, the primary difference between the two is how they prove the validity of their transactions. Optimistic rollups compress and post all of their transaction data, assume that it is valid, and use fraud proofs in the case of a challenge. Zk-rollups, on the other hand, simply post validity proofs to the L1 to prove the validity of each transaction. This means that, while optimistic rollups have to post all of their transaction data to the L1, which is expensive and time-consuming, zk-rollups only have to provide validity proofs. **Both types of rollups post this data to Ethereum using "calldata."** Calldata is another name for the data passed along with an Ethereum transaction that enables information exchange. However, we should note that calldata is neither optimized nor scalable enough to meet the needs of L2s posting their data back to Ethereum. In addition, calldata lives permanently on-chain, which contributes to state bloat, i.e., imposes a storage load on Ethereum full nodes. We should note that since EIP-4844 recently went live, most major rollups switched to posting their data on a combination of calldata and 'blobs', which has started reducing this burden. (EIP-4844 is explained below.)

When rollups post their transaction data (or validity proofs) to the L1 for settlement, they are essentially using the L1 for DA. For example, when Arbitrum One posts transaction data to Ethereum, they are using the **native Ethereum DA layer** (although we should note that calldata was never initially intended to be used for DA purposes but has been more of a solution in the absence of dedicated EIP-4844 blobspace).

However, due to the resource constraints of Ethereum, namely chain congestion and limited space for data storage, L2 throughput can be limited, and extra costs can arise. **To post data to Ethereum, L2s have historically been subject to the same fee market, block size constraints, and block times as regular transactions** (prior to EIP-4844). In addition to imposing a considerable load on full Ethereum nodes that must download all of this data, L2s are also subject to spiking fees, which can happen during periods of peak demand. As previously mentioned, over 90% of L2 fees can sometimes be driven solely by DA costs.

The Data Availability Problem

Data availability refers to **the confidence a user can have that the data required to verify a current block is truly available to all network participants**[1]. Following on, the data availability problem refers to the challenge of proving this fact to the entire network without requiring all nodes to download all of the data. The idea is that any independent verifier should be able to download the required transaction data to verify that a block is valid.

> *"Data availability refers to the confidence a user can have that the data required to verify a current block is truly available to all network participants."*

We should note that **DA is not the same as data retrievability,** i.e., DA does not imply a historical database of a blockchain's transaction data; it is more akin to a short-term guarantee that you can download the data and that it is available. We can consider the mental model of publishing data to a website, based on which we can assume that enough people saw the data or were able to download it. However, publishing something on a website is not a guarantee that it will be available to download one year later. This is what DA is about – it is about the strength of the guarantee that the data was published and made available for everyone to see or download.

> *"We can consider the mental model of publishing data to a website, based on which we can assume that enough people saw the data or were able to download it. However, publishing something on a website is not a guarantee that it will be available to download one year later. This is what DA is about – it is about the strength of the guarantee that the data was published and made available for everyone to see or download."*

Although the full transaction data is necessary for independently verifying blocks, the DA problem tells us that requiring all nodes to download all transaction data is a barrier to scaling. Solutions to the DA problem aim to provide sufficient assurances that the full transaction data is made available for verification to network participants that do not download and store the data for themselves.

Celestia, Avail, and EigenDA have emerged and aim to allow data to be stored cheaply, while guaranteeing DA, thereby creating a specialized DA layer and giving L2s options outside of the native Ethereum DA layer.

Ethereum's Dencun Upgrade and EIP-4844

Ethereum's Dencun hardfork went live on March 13, 2024, and consisted of several upgrades, including the much anticipated EIP-4844 (also known as "Proto-Danksharding").

EIP-4844 was a DA-focused upgrade and introduced a new transaction type to Ethereum called a "blob-carrying transaction," which is akin to a regular transaction but also carries an extra piece of data called a "blob." Blob-carrying transactions are **not executed by the EVM** but in the consensus layer instead and are **only stored temporarily** rather than permanently like regular Ethereum transactions. Blobs are also priced in a **separate gas market** and are thus not competing with the Ethereum L1 gas market. **Rollups can choose to post their data on these blobs rather than on the Ethereum L1 via calldata, as they have historically done.** Overall, the combination of blobs' temporary nature and separate gas market, alongside other features, helps create an optimized and cheaper solution for L2 rollups to publish their data for DA purposes.

Figure 3: Difference between Ethereum L1 block space and EIP-4844 blob space

Feature	Ethereum L1 block space	EIP-4844 Blob space
Seen by all nodes?	Yes	Yes
How long is data stored?	Permanent (forever)	~18 days
Visible to EVM?	Yes	No
Gas market	Main Ethereum L1 gas market	Separate blob gas market

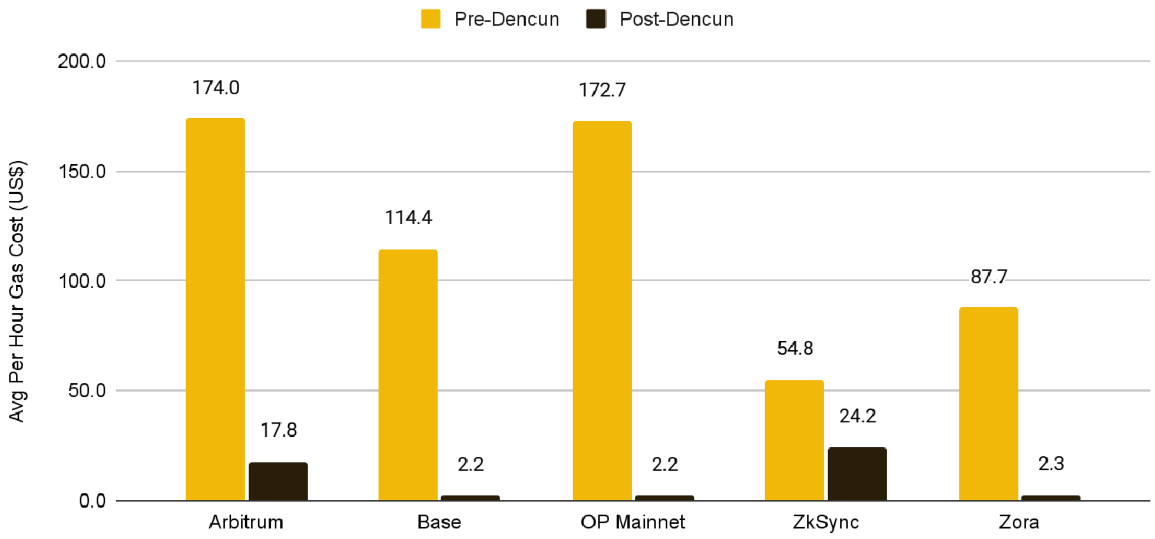
Source: EIP4844.com, Binance Research

Impacts

As expected, since the implementation of EIP-4844, the cost of Ethereum's L2 rollups has significantly shrunk, impacting both user fees and L2 profitability.

When we consider the cost of posting data on Ethereum, the **average hourly cost for some of the largest L2 rollups is down from over US$270 to under US$25**. Given that an L2 rollup's profit is essentially the transaction fees minus the cost of posting data, this has a direct impact on L2 profitability.

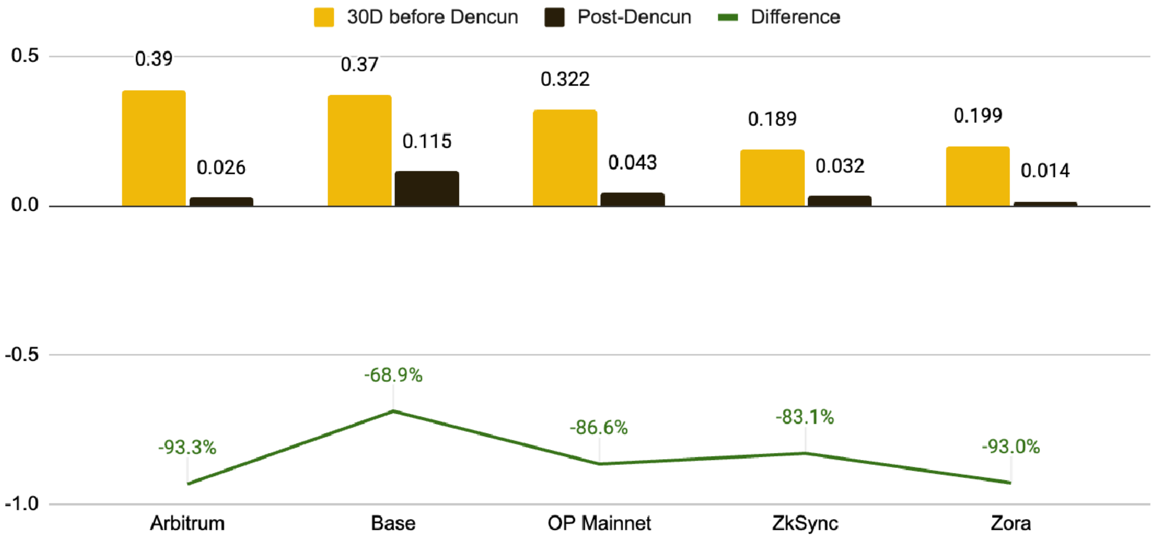
Figure 4: The cost of posting data to Ethereum has seen an average decrease of 72% across the top Ethereum L2s



Source: Dune Analytics (@21co), Binance Research, as of April 9, 2024

As previously highlighted in Figure 2, a significant proportion (up to 90%) of major rollups' costs were related to DA costs. Given EIP-4844 materially lowered these, the **transaction fees to interact on the rollup have also decreased quite significantly.** Ethereum L2 users are now seeing fees that are down by up to 90% compared to pre-Dencun.

Figure 5: L2 transaction fees have decreased by 50-95%



Source: Dune Analytics (@21co), Binance Research, as of April 9, 2024

While the above two charts paint a relatively positive picture of EIP-4844's impacts so far, when we consider transaction throughput (measured by transactions per second), a more nuanced discussion arises.

We must consider the potential **limitations of Ethereum's current and short-term scalability**. For instance, even under EIP-4844's ideal state of full Danksharding, the **theoretical TPS achievable with dedicated DA layers is significantly higher**[(2)], indicating that L2s opting to use solutions like Celestia could benefit from cost efficiencies and potentially higher margins.

So, how do the likes of Celestia, EigenDA, and Avail promise such levels of transaction throughput? In the next section, we take a look under the hood.

4 Key Projects

In the next two sections, we will focus on the technical underpinnings of DA solutions. Before we dive in, let's compare the dedicated DA layers featured in this report in Figure 6 below.

Figure 6: Comparing notable DA projects in the market

	Celestia	EigenDA	Avail
Main advantage	First mover, **pioneer** in the space	Likely more **scalable**, by having lower safety threshold (supermajority rather then the onerous 1-in-n honest nodes)	"Celestia with **faster** DA **finality**"
Live?	Mainnet	Testnet	Testnet
Theoretical scalability	**High** Constrained by high storage burden of full storage nodes	**Very high** Constrained by communication overhead of dispersal protocol and Ethereum's latency for storing block headers	**Medium-high** Likely similar to Celestia, although may be a little lower due to KZG
DA guarantee safety threshold	**Very high** DA layer: improves with more light nodes. **1-in-n** (potentially 0-in-n) honest full storage nodes. Consensus layer: **1-in-n** (due to fraud proof mechanism)	**Regular** Depends on parameters. Likely requires majority or supermajority honest nodes, which are secured by EigenLayer	**Very high** **0-in-n.** No need for fraud proof mechanism due to KZG validity proofs. Similar to Celestia otherwise
Liveness / Censorship resistance threshold	TVL of staked tokens	Security from EigenLayer restaking	TVL of staked tokens
DA "finality"	**Medium** Requires challenge period	**Fast** Uses KZG validity proofs, but Ethereum latency for block header finality	**Very fast** Uses KZG validity proofs
Data Storage Burden	**Large** Full storage nodes: replicate full dataset Validators/Full consensus nodes: block hashes only	**Medium** Network nodes (EigenDA): data shards Ethereum nodes: block hashes only	**Large** Nodes store full dataset

Communication Burden	**Medium** Communication of full data for storage nodes	**Medium-High** Communication overhead of disperser protocol	**Medium** Same as Celestia
Decentralization Level	**High**	**Medium** EigenLayer requires enough Ethereum nodes to opt in. Centralization of EigenLayer at this early stage.	**High**
Chains served (for settlement)	**Any** Rollups' own or use existing settlement layers. Blobstream a pre-built solution	**Any** Ethereum likely a natural choice	**Any** Likely greater support with Nexus
Maturity	**Medium** Solution produced in 2018. Mainnet launched	**Low** Solution produced in 2021	**Low** Reference document published in 2021
Delivery format	Layer 1 blockchain (Tendermint)	Network secured via EigenLayer restaking	Layer 1 blockchain (likely Substrate)

Source: Project whitepapers, Binance Research, as of 26 March, 2024

Protocol designs are changing constantly, especially for protocols that have yet to launch. The information here is based on currently available information in the public domain, which may change quickly.

Celestia

Summary

Celestia is the first dedicated project aiming to solve the scalability problem by providing a DA layer. Celestia is designed simply to make data available without checking if transactions are valid. The team pioneered this idea in their research paper[(3)] published in 2018/19 with Vitalik Buterin. Having a high-performance DA layer removes a key constraint for rollups to reach very high throughputs.

At a high level, Celestia consists of the 1) the consensus layer, 2) data availability layer, and 3) bridge nodes that connect them.

Consensus Layer

The consensus layer is a Cosmos appchain. Consensus nodes accept data from clients (requests to store data), process the data, and arrange these into blocks. A selected validator produces and publishes the block. Validators consider a block valid if the underlying data is indeed available. The block includes transactions from clients and extended data (see the erasure code section).

A compromised consensus layer can produce incorrect erasure encoding. To counter this issue, full nodes can submit fraud proofs if this happens. This feature is why some refer to Celestia's design as "optimistic"; like optimistic rollups, there is a challenge period, and if no fraud proofs are submitted, the block is considered valid. Therefore, Celestia does not require a majority of the consensus to be honest to guarantee data availability.

While Celestia can resist some attacks from a compromised consensus layer, ultimately, such a situation will eventually compromise the system.

Data Availability ("DA") Layer

The DA layer contains the core functionality of Celestia, implementing the key innovation in Celestia's research paper. This is where information is actually stored and guaranteed to be available to clients (e.g., rollups).

There are two nodes on this layer: full storage nodes and light nodes. At a high level:

- **Full storage nodes** ("FSNs") actually store the data and allow other parties to download them.
- **Light clients** ("LCs" or "light nodes") ensure that full storage nodes uphold their responsibility.

Unlike most blockchain designs, Celestia's LCs directly contribute to security in Celestia's DA layer. They do so via data availability sampling ("DAS"). LCs sample random data chunks from FSNs. If FSNs cannot send a valid response, LCs submit a fraud proof.

DAS allows LCs to gain high confidence that full data is available by only downloading small chunks of data. For each sample, there is a roughly 25% chance of detecting whether the full original data is not available. After only 15 samples, there is around a 99% probability of detecting any missing data (we explain why in further detail in the DAS section). The system becomes more secure with more LCs.

Theoretically, due to DAS, there only needs to be one honest FSN to have DA. That said, LCs need correct block headers from consensus nodes, which are secured by the Celestia chain consensus and a fraud-proof mechanism.

Key tech primitives used: erasure codes and DAS

EigenDA

Currently in its testnet phase, EigenDA is another dedicated data availability solution. The end product for clients (i.e., rollups) is similar, but EigenDA guarantees DA differently. First, EigenDA is planning to be an actively validated service ("AVS") on EigenLayer. Being an EigenLayer AVS means that the crypto-economic security is dependent on EigenLayer, which has slightly different game theory dynamics than a PoS blockchain.

Secondly, **instead of DAS, EigenDA is being built around a dispersal protocol.** Unlike DAS, a dispersal protocol can guarantee data availability as long as the threshold number of nodes is honest. The team released a paper[(4)] describing a protocol called ACeD. The design appears to have shifted a little since.

Key tech primitives used: erasure codes, dispersal protocol, and KZG (in new design)

Avail

Avail's design is more similar to Celestia (at a high level) than EigenDA. It is also designed to be an independent L1 blockchain and utilizes DAS in a mostly similar way to Celestia.

In some sense, Avail has benefitted from being a later entrant; its design can be seen as an improvement over Celestia's current design. A **key difference is that it uses KZG to produce proof that the block producer's erasure codes are correct.** This means that LCs can perform DAS immediately after a block is produced, rather than waiting for a challenge period to be over. Avail thus achieves faster DA finality. The downside is that it takes more computation to produce these KZG proofs.

The use of KZG makes Avail[(5)] potentially more in line with zk-rollups. Zk-rollups achieve faster finality than optimistic rollups. Also, the native use of KZG could potentially result in some optimizations as many zk-SNARKs protocols utilize KZG.

Key tech primitives used: erasure codes, DAS, and KZG

A Note on Danksharding

Danksharding is likely to utilize both KZG and DAS, making it somewhat similar to Avail on the surface. However, Danksharding is not meant to be a dedicated DA layer, unlike the other protocols, but instead a DA layer built into the Ethereum base layer. This would make Ethereum both a settlement layer and a DA layer.

Note that Proto-Danksharding is already live, but it mostly does not implement the primitives we discuss here. It is a temporary solution before Danksharding, introducing a non-permanent, hence cheaper, block space to Ethereum.

It is important to note that neither Proto-Danksharding nor Danksharding are "sharding" anything in the traditional sense.

5 Key Tech Primitives

The goal of this section, the main body of this report, is not to extensively describe the architecture of specific protocols but to dive deeper into four important technological primitives: erasure codes, data availability sampling ("DAS"), dispersal protocols, and KZG. We will see from first principles the reasons for key product differences between Celestia, EigenDA, and Avail.

Primitive 0: Erasure Codes, Used by All Protocols Featured in This Report

Erasure codes are used by all protocols featured in this report and are also the foundation for two other primitives.

Erasure codes (as well as error-correcting codes) are **extensively used in communication and data storage**. Suppose you make a call from your mobile phone. Without erasure coding, any data lost during transmission could result in noticeable glitches or even cause the call to drop. With erasure coding, lost data can be reconstructed.

This is done by adding data redundancy. A 1kb dataset is extended to, say, 2kb. For an optimal erasure code, any random 1kb of data can be lost in transmission, and the original data can still be recovered.

Erasure codes are also used in data storage to improve resilience to data loss or corruption.

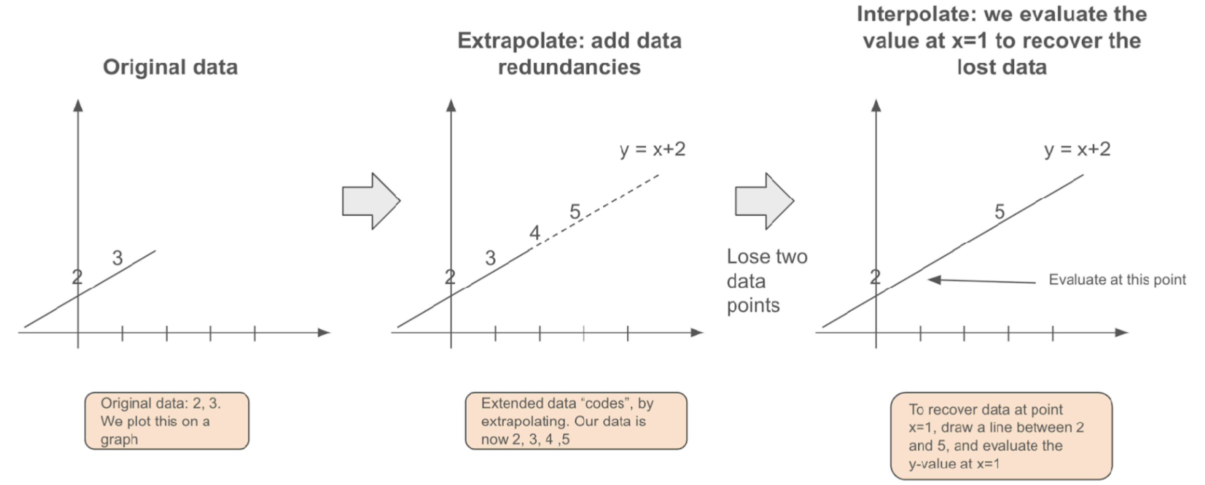
How is this related to data availability?

Erasure codes are used in DA protocols, not for making data resilient per se. It is used for its many useful properties, which you will see in later sections. For example, when used in combination with DAS, it results in a very efficient method for light clients ("LCs") to ensure full nodes are indeed making data available.

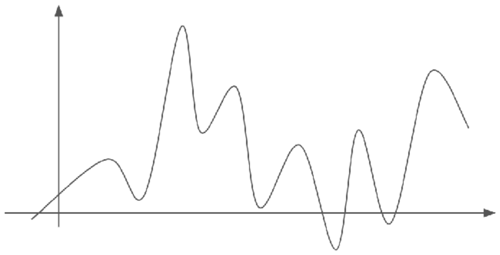
For now, it suffices to understand the basic concept of what erasure codes are.

Technical Explanation (for the math-inclined)

Erasure codes add redundancy to data. In an oversimplified example, take a two-element dataset {2, 3}. We first represent this data on a plane, where the x-axis is the position of the data (e.g., incrementing from 0, 1, 2, ...) and the y-axis represents the data value (2 and 3 in this case). We convert this two-element dataset into a four-element dataset simply by drawing a straight line that connects the two points.


Original data
3
2
Original data: 2, 3. We plot this on a graph
Extrapolate: add data redundancies
y = x+2
5
4
3
2
Extended data "codes", by extrapolating. Our data is now 2, 3, 4 ,5
Lose two data points
Interpolate: we evaluate the value at x=1 to recover the lost data
y = x+2
5
2
Evaluate at this point
To recover data at point x=1, draw a line between 2 and 5, and evaluate the y-value at x=1

Indeed, Erasure codes such as Reed Solomon work in this manner. The main difference is that it uses polynomials, not straight lines, because arbitrary data will not fit on a straight line. There is a unique polynomial of lowest degree that crosses an arbitrary set of points. It can be found using some fancy math (e.g., Lagrange interpolation). Using polynomials, the chart will look more like this:



The concept is the same as our straight line extrapolation and interpolation. We can derive the polynomial equation without the full dataset. If, for example, the data at x = 5 is missing, we can evaluate the polynomial at x = 5. The y-value is the recovered missing data.

Primitive 1: Data Availability Sampling ("DAS"), Used by Celestia, Avail, and Danksharding

Data availability sampling enables very high confidence in data availability with minimal hardware requirements.

The aim of DAS is to allow resource-constrained LCs to gain confidence that full storage nodes ("FSN") are indeed making data available.

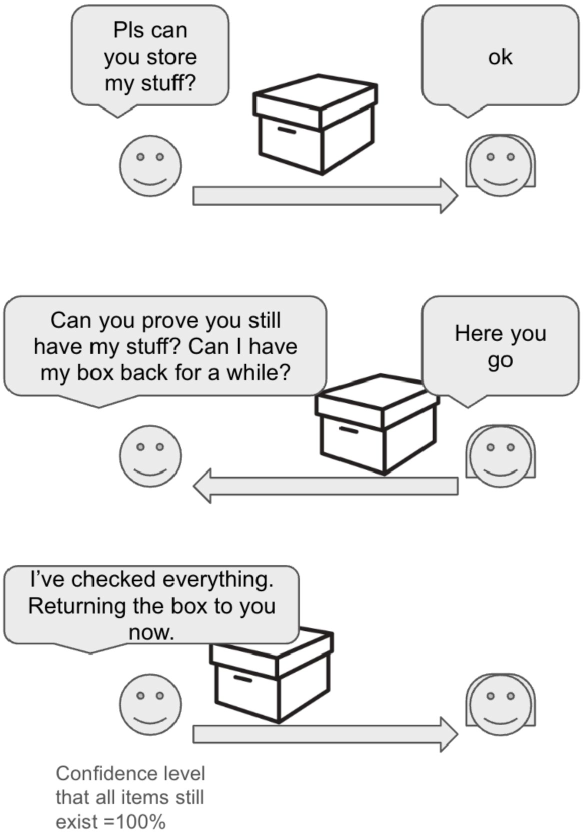
Let's imagine this scenario: Bob has too many items (1,000 items to be precise), so he asks his friend Alice to store his items. Despite Alice's kindness, Bob is paranoid and doesn't trust her. He insists on having a mechanism to check that Alice still has all his items.

How can we design such a system?

Design 1: A Naïve Design

Bob (on the left) periodically requests that Alice (on the right) send the full package of 1,000 items back to him. He then checks to see if everything is still there. Once he's satisfied, he sends the package back to Alice for long-term storage. Doing this will give Bob the comfort he needs, because if Alice throws anything away, Bob will be able to see that the item is missing.

Of course, there are **several problems with this system**. Firstly, Bob lacks space. He needs to find a temporary space in his hallway to hold all his items while he checks them. Secondly, they need to ship many items, so their courier bills will be high.


Pls can you store my stuff?
ok
Can you prove you still have my stuff? Can I have my box back for a while?
Here you go
I've checked everything. Returning the box to you now.
Confidence level that all items still exist =100%

In a DA scenario, Alice is analogous to the FSN and Bob the LC. Design 1 works but requires the LC to temporarily procure more memory to store all items (e.g., short-term renting a cloud service). It also has high communication overhead because the FSN sends the LC the entire dataset. Note that in a DA protocol, the LC does not need to "send back" the data to FSN, unlike in our analogy.

Design 1 shows us the purpose and basic interactions that happen in DAS.

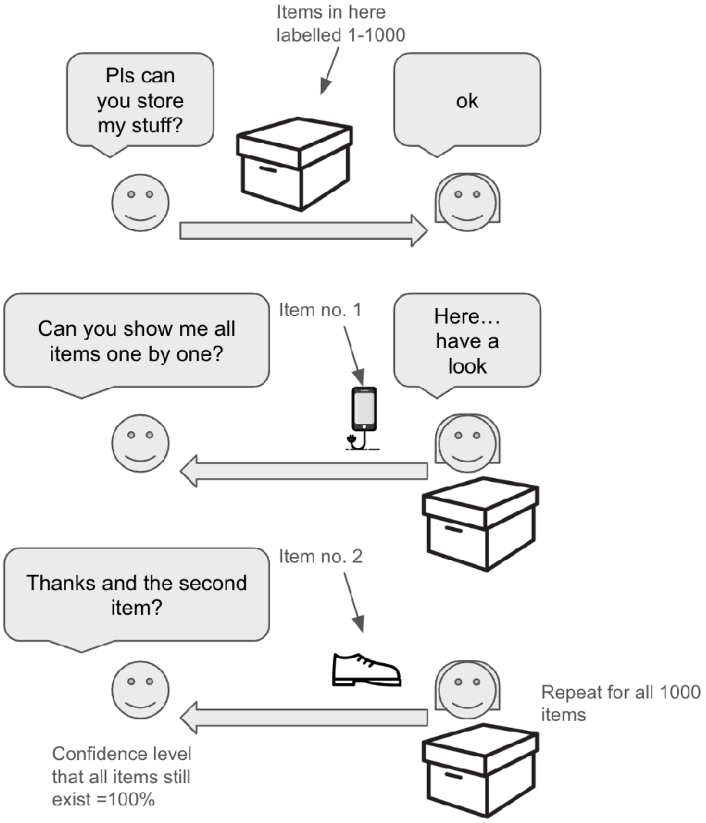
It is highly inefficient, though. Can we do better?

Design 2: A Slightly Better Naïve Design

Bob asks Alice to show him all 1,000 items, one by one. As before, after this procedure, Bob has 100% certainty that Alice has kept all the items.

This alleviates the storage problem. The LC no longer needs to procure temporary storage, but network communication remains high.

Design 2 **shows us how working with a chunk of data at a time almost eliminates storage requirements** for the LC.


Items in here labelled 1-1000
Pls can you store my stuff?
ok
Can you show me all items one by one?
Item no. 1
Here… have a look
Thanks and the second item?
Item no. 2
Repeat for all 1000 items
Confidence level that all items still exist =100%

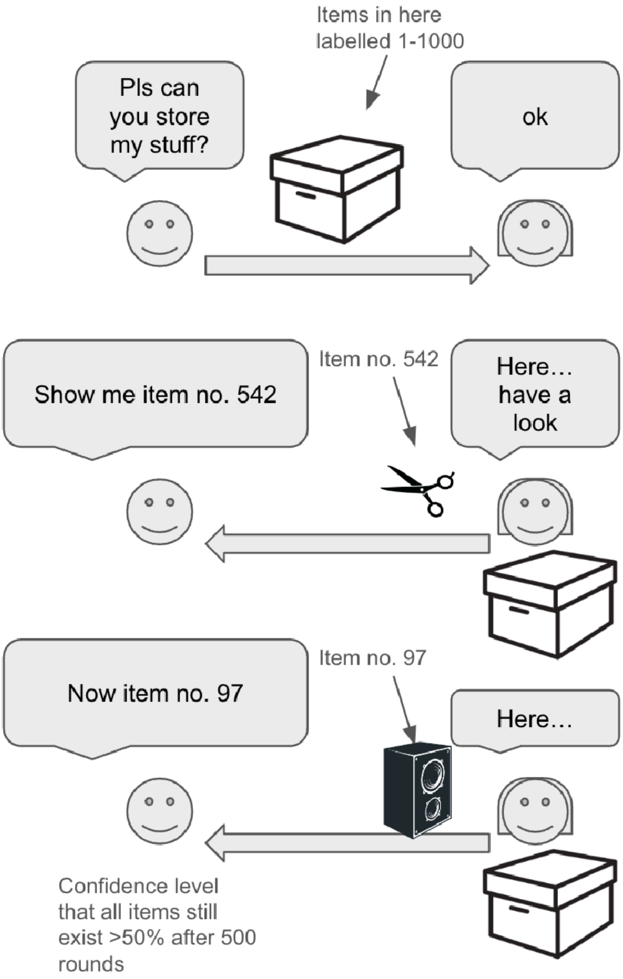
Design 3: Basic Sampling

Instead of having proof (100% certainty), we design a system where Bob can be highly certain that Alice is doing her part. We modify Design 2 such that Bob requests Alice send him a list of x items rather than all 1,000. Alice cannot know in advance what that list is. Otherwise, she can throw away all other items and keep only the ones she knows Bob will sample.

We illustrate how this looks if x = 500.

Let's say Alice produces a valid response. **How confident is Bob that Alice has ALL his items now?**

Bob sampled 50% of the items, so a starting point is to say he is 50% confident. However, due to the random nature of sampling, it turns out that Bob is probably more than 50% confident. **In fact, Bob is at least 50% confident — he could be more confident depending on his beliefs about Alice's behavior.**


Items in here labelled 1-1000
Pls can you store my stuff?
ok
Show me item no. 542
Item no. 542
Here... have a look
Now item no. 97
Item no. 97
Here...
Confidence level that all items still exist >50% after 500 rounds

Specifically, he is exactly 50% confident if he believes that the only way Alice could misbehave is to throw away one item. His confidence is higher than 50% if he believes Alice could have thrown one or more items away.

Technical Explanation (for the math-inclined)

Let’s drastically simplify the numbers. Assume Alice stores 4 items and Bob samples 2 items. D represents the scenario where Alice is missing items but gets away with it.

If Alice produces a valid response (i.e., successfully shows the items that Bob requests), then Bob’s confidence, C, that Alice still has all his items is:

$$C = 1-P(D)$$

To calculate $P(D)$, we can use combinations. The number of possible ways Bob can sample 2 items out of 4 is:

$$C_2^4 = 6$$

Let’s assume Alice throws away 1 item. In this scenario, the number of ways Bob can choose 2 items and fail to catch Alice cheating is $C_2^{4-1} = C_2^3 = 3$. Therefore, the probability she gets away with it is:

$$P(D) = C_2^3 / C_2^4 = 3/6 = 50\%$$

Therefore, Bob’s confidence level is:

$$C = 1-P(D) = 1 - ½ = ½ = 50\%$$

But Alice could have also thrown away 2 items and gotten away with it. If we rerun the above calculations, we get:

$$C = 1-P(D) = 1 - C_2^2 / C_2^4 = 1-⅙ = \sim 83\%$$

Alice could not have possibly thrown away 3 items and gotten away with it. If she did, she would have 1 item left. Bob samples 2 items and has a 100% chance of catching her misbehavior.

Therefore, Bob’s confidence that Alice in fact has all items is 50% $\leq C \leq$ 83%. If Bob has no opinion on Alice’s possible behavior, we could assign an equal chance that Alice may have thrown away either 1 or 2 items and got away with it. His overall confidence is thus:

$$C = (½ + ⅚)/2 = ⅔ = \sim 67\%$$

To recap, if Alice is supposed to store four items, Bob chooses two randomly and asks that she show them. Alice successfully does so. Bob is now 67% sure that Alice still has all four items with her.

Bob **sampled 50% of the items and got 67% confidence**. Notice the confidence level is higher than the percentage sampled. This is because Alice cannot predict Bob's random sampling choices. It is the **power of random sampling.**

Design 3 demonstrates that sampling is more efficient. If we can live with confidence below 100%, we can reduce resource requirements for the LC.

"Bob sampled 50% of the items and got 67% confidence. Notice the confidence level is higher than the percentage sampled. This is because Alice cannot predict Bob's random sampling choices. It is the power of random sampling."

Design 4: Using Erasure Codes

Design 3 is still limited. With 50 items and a sample size of 25, Bob's confidence will be between 50% and 96% (where 96% is the result of assuming equal probability, as we did above). In reality, his confidence would be closer to 50%, because Alice is more likely to discard only a few items because it is easier to get away with that. Bob needs to sample about 80% to have approximately 90% confidence. In crypto, 90% confidence is low. How would you feel if, for every transaction you make on chain, there is a 10% chance of your tokens disappearing?

Also, if Bob samples 80%, he might as well just sample everything and gain 100% confidence. This would simply be Design 2.

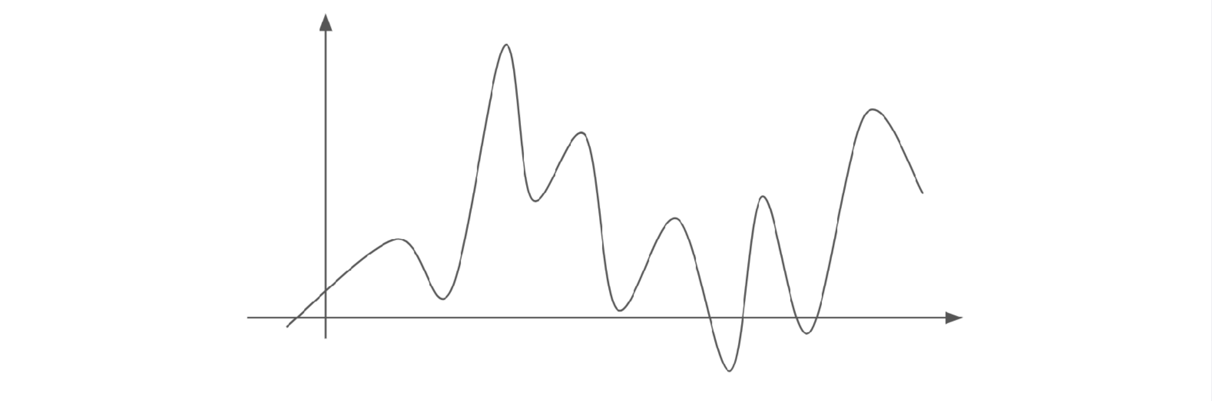
Design 3 is not workable in its current form. **But can we improve our system? Can we sample 50% of the items to gain 99% confidence, for example? How about sampling 1% to gain 99.99% confidence?**

The **game-changer is a system where sampling a fixed number rather than a percentage results in a certain level of confidence.** There is a system where you can sample 14 items to gain 99.99% confidence. Notice the difference: we've moved from percentage of items to number of items, regardless of the total size. If Bob has 1,000 items, he samples 14 items (1.4%) to gain 99.99% confidence that all items still exist. If he has 1,000,000 items, he still samples 14 (0.0014%) items to gain 99.99% confidence.

14 samples will give over 99.99% confidence (1-in-10,000 error). At 20 samples, it is 1-in-a-million. Using erasure codes is extremely efficient.

Technical Explanation (for the math-inclined)

We previously described erasure coding. The equation of the unique polynomial of lowest degree can be found, such that its x-values are the position of the data element (0, 1, 2,...) and its y-values are the corresponding values in the dataset. It may look something like below.



We then extend the dataset by evaluating points at different x values. For example, if our dataset size is 1000, we have x-values {0, 1, ..., 999}. We then evaluate the y-values at points {1000, 1001, ..., 1999}, which doubles the dataset size. This double-sized dataset (all the y-values) is the coded data, which is stored.

Later on, as long as we still have any arbitrary 1000 out of the total 2000 data points, we will be able to reconstruct the same polynomial equation and derive the original dataset.

The key insight is that the original data can be reconstructed as long as 50% of **any** of the extended data points are available. Therefore, in order for data to become unavailable, the storage nodes need to discard >50% of the data points. If 50% of the data is missing, the **LC has a 50% chance of detecting this each time it samples the data**. Therefore, just one sampling procedure gives a 50% confidence level to the LC.

We have successfully arrived at a design where the LC can sample very few items from the FSN and gain high confidence that the FSN has the complete range of data available.

Design 5: 2D Reed-Solomon, As Used by Celestia

The design above already solves our data sampling problem. So why does Celestia go even further?

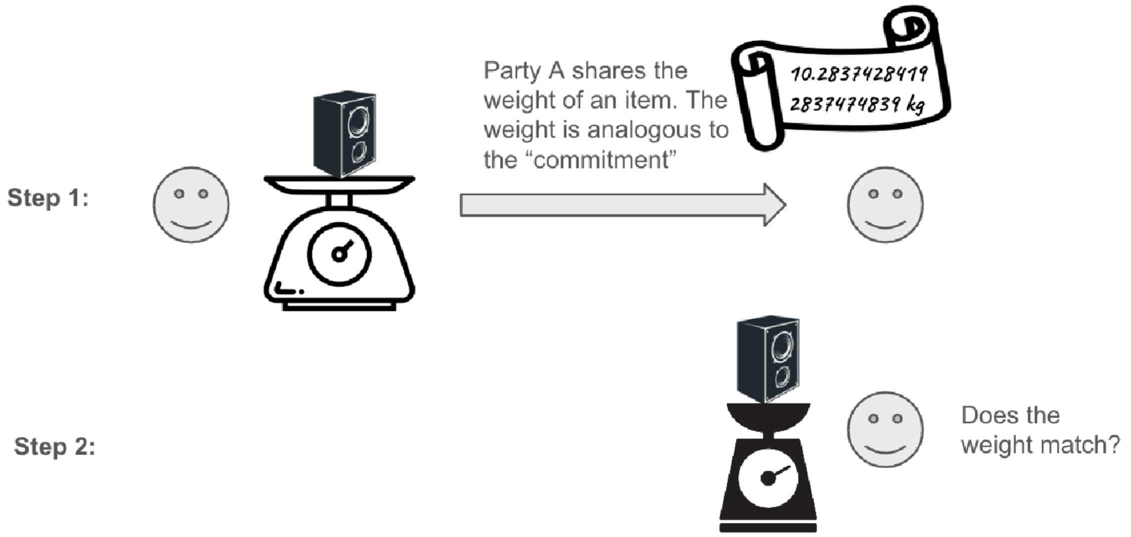
It is mostly to solve other problems that arise in practice. So far, we have assumed Bob is familiar with his list of 1,000 items. But DAS, in fact, resembles a situation more akin to the following:

Bob asks Alice to store his items. Carol then checks whether Alice still has them. However, Carol does not know what items Bob should have.

How does Carol do this check when she doesn't even know what the items should be?

This is done using **Merkle commitments and proofs**. The entire block of items is first committed, i.e., compute the Merkle roots. With these roots, Carol can request a random item from Alice, say item 549. Alice sends a Merkle proof to Carol, which includes revealing the item. Carol can verify the Merkle proof against the Merkle root. A successful verification is a cryptographic proof that the item is indeed item 549.

This is a similar process that LCs use to retrieve data from a full node and have cryptographic confidence that it is valid. If you understand Merkle proofs, this should make sense. If not, we offer an imperfect analogy of using a fine weighing scale in this diagram:


Party A shares the weight of an item. The weight is analogous to the "commitment"
10.2837428419 2837474839 kg
Step 1:
Step 2:
Does the weight match?

Assume people have access to weighing scales so fine that it's not feasible to find a second item with exactly the same weight reading. Alice can commit to her item (the audio speaker in the illustration) and share the weight (the commitment) with Bob. At a later time, Alice sends the speaker to Bob. Bob is unsure if it is the exact speaker that Alice showed him before. In order to check this, he weighs the speaker. If the weight matches, then he can be confident that it is the same speaker.

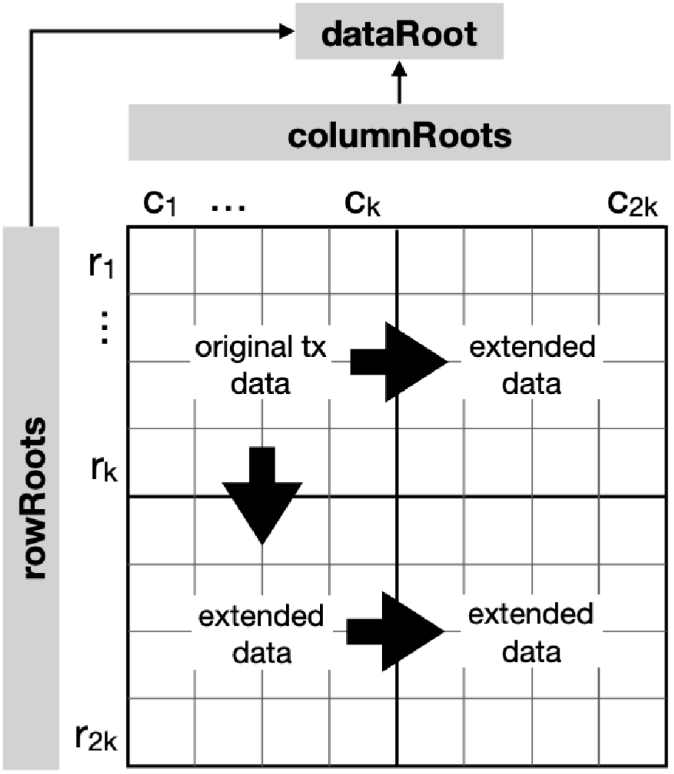
This analogy actually describes a hash function like SHA256. Merkle trees are more complex. They can commit to large datasets by sharing just a single hash.

We've now reached the point commonly discussed. **Celestia uses 2D Reed Solomon ("2DRS").** Design 4 describes 1D Reed-Solomon ("1DRS"). The concept is the same, but the main advantage of using 2DRS is to reduce proof sizes and verification time complexity. The downside is much larger root hash sizes. Additionally, LCs need to sample more times (approximately double) to gain the same confidence level that data is available.

Technical Explanation (for the math-inclined)

In a 2DRS, the LC obtains the data root, column roots and row roots. When the LC requests a data point, the FSN can choose to provide the Merkle proof either along the row or column. If the FSN provides a row (column) Merkle proof, the LC can verify it against the row root (column root) that it obtained from the consensus nodes.

Celestia's 2D diagram



Source: Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities (Link)

A 1DRS only needs the data root hash. Conversely, the root size in 2DRS is much larger because it includes every row and column hash, as well as the data root hash. It is worth the sacrifice as fraud proof sizes drop from a size similar to downloading all data to O(√n log(√n)), which is around the length of the square, which is significantly smaller than the area. It also lowers the proof size and verification cost from log n to log √n, a small saving.

With 2D Reed Solomon, sampling once gives ~25% confidence, rather than 50% with a 1D Reed Solomon. It's a small trade-off, as sampling 20 times still gives over 99.5% confidence. For a 1-in-a-million confidence level, asymptotically (i.e., assuming a large dataset), LCs need to sample ~50 times (compared to 20 in the 1DRS case).

The upshot is that, compared to 1DRS, using 2DRS reduces the computation burden on LCs, particularly for producing fraud proofs. The downside is larger block header sizes.

[Celestia's use of] 2DRS reduces the computation burden on LCs, particularly for producing fraud proofs.

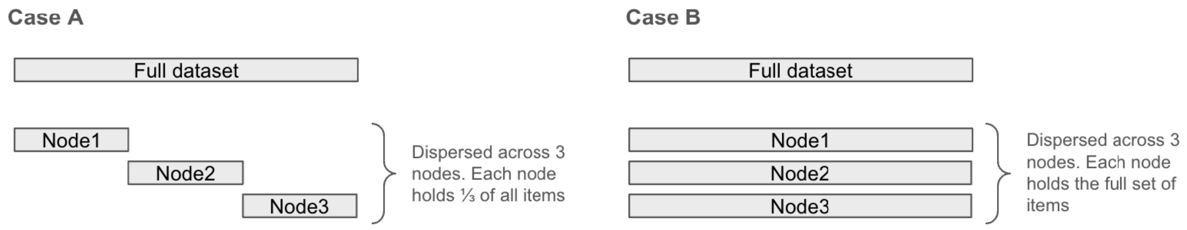
From these first principles, we've seen how and why Celestia's DAS works. Avail's DAS and likely Danksharding's DAS have different designs. For example, Avail extends data only along the vertical axis. However, the fundamental concepts are the same.

Primitive 2: Dispersal Protocols, Used by EigenDA

A dispersal protocol can improve scalability while still guaranteeing data availability.

Dispersal protocols disperse a dataset to multiple nodes, such that each node does not store the entire dataset, but the network collectively holds the full dataset. A benefit is lower hardware requirements for nodes.

There is a trade-off between data robustness and total storage requirements. Let's explore the two extremes of this spectrum:


Case A
Full dataset
Node1
Node2
Node3
Dispersed across 3 nodes. Each node holds ⅓ of all items
Case B
Full dataset
Node1
Node2
Node3
Dispersed across 3 nodes. Each node holds the full set of items

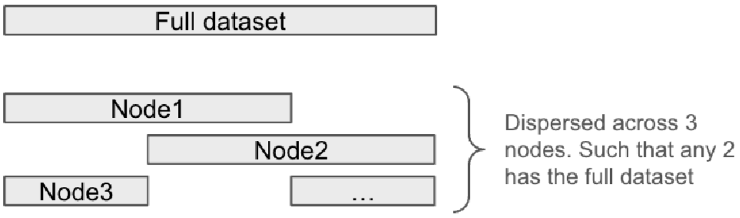
These two methods are simple but not very useful. Case B can hardly be called a dispersal protocol. Things get more complicated in the middle.

Let's define two important measures:

- **Byzantine threshold:** the percentage of malfunctioning or dishonest nodes the system can withstand. The Byzantine threshold is 0% in Case A and 67% (more precisely, 1-in-n) in Case B.
- **Storage blowup:** the increase in total storage space required by all the nodes. The storage blowup is 1x in Case A and 3x (more precisely, O(n)) in Case B.

As you can see, there is a trade-off between these two measures. Cases A and B sit at the extreme ends of this tradeoff. (Another important measure is the communication complexity, but we will ignore that in this description for simplicity.)

Next, we attempt to design a protocol that makes a tradeoff between these two measures. Here is a simple protocol where the full data can be recovered as long as <33% of the parties are Byzantine:


Full dataset
Node1
Node2
Node3
...
Dispersed across 3 nodes. Such that any 2 has the full dataset

Notice that if Node 1 misbehaves, Nodes 2 and 3 combined still have the full dataset. The same is true if either Node 2 or Node 3 misbehaves; the other two combined always have the full dataset.

In our simple protocol, we have a 33% Byzantine threshold and a storage blowup of 2x. Since we have 3 nodes, each node holds ⅔ of the full dataset, saving space for the individual nodes.

This may seem adequate, but there are some problems. First, it doesn't scale. What if we had 30 nodes instead? A simplistic way to do this is to duplicate what we had 10 times. The storage blowup will be much larger, perhaps more in the order of 20x. Second, it has a 33% Byzantine threshold only in the worst-case scenario, where all nodes that fail happen to hold the same subset of data. This makes it inefficient.

It is possible to achieve protocols that scale better.

AVID is a protocol introduced about 20 years ago that EigenDA's dispersal protocol is partly based on (EigenDA team published a paper[6] on ACeD which builds upon AVID and CMT. However, it has made changes in its current design. For example, it will likely use KZG rather than CMT.) AVID achieves much better trade-offs, and importantly, it scales well.

AVID-RBC, one of AVID's variants, is able to achieve a 33% Byzantine threshold with a storage blowup of 1.5x (plus a constant overhead), regardless of the number of nodes. In our example with 3 nodes, AVID-RBC achieves 1.5x vs. our 2x. More importantly, the storage blowup is 1.5x + a constant, regardless of how many nodes there are, unlike our simple protocol.

Like DAS, AVID achieves this by using erasure codes.

Technical Explanation

We explain a simplified version of AVID as a way to understand this intuitively:

Assume there are n nodes. For a t = n/3 Byzantine threshold, we need to increase the data size by 1.5x via erasure codes. In other words, for a (k, n)-erasure code, $n/k = 1.5$. (Note that by convention, k is the size of the original message and n is the size of the coded message).

Next, we split the codes into n chunks and, after creating a hash signature for each, require each node to hold one of the n chunks.

We can see how, by doing this, as long as ⅔ nodes are honest and functional, the original data can be recovered via erasure codes. The storage blowup, assuming a reliable broadcast (RBC), is therefore:

$$\frac{n}{n-t} + O(1)$$

This is 1.5x + a constant, when t = n/3, regardless of how large n is.

Compared to AVID, EigenDA's dispersal protocol has a slightly less efficient storage blowup (it scales logarithmically with the block size) but has much better communication complexity.

With a dispersal protocol, as more nodes join the system, the storage burden per node drops. This makes EigenDA very scalable, at least from a storage hardware requirement perspective.

With a dispersal protocol, as more nodes join the system, the storage burden per node drops. This makes EigenDA very scalable, at least from a storage hardware requirement perspective.

EigenDA relies on its dispersal protocol both to reduce hardware requirements and as the basis of its DA guarantees. The protocol parameters can be set such that the protocol can resist up to ⅓ faulty nodes, making it resemble a typical PoS consensus safety threshold. DA is guaranteed as long as the threshold percentage of nodes is honest. Therefore, DAS is not used.

Primitive 3: KZG, Used by Avail, EigenDA, and Danksharding

KZG has useful features that Avail uses to speed up DA finality.

KZG is a cryptographic commitment scheme with several useful properties. Just like Merkle trees, a common commitment scheme, KZG allows Party A to commit to a value (e.g., a dataset) and later use that commitment to prove specific elements (e.g., specific points in the dataset) belong to that committed dataset.

Unlike Merkle trees, KZG uses polynomials, a mathematical expression. Polynomials have many useful features, which makes KZG better for certain use cases.

Advantages of KZG over Merkle trees:

- Proof sizes and verification time for KZG are constant O(1), while they scale logarithmically for Merkle Trees, i.e., O(log n).
- KZG is also partially homomorphic, a mathematical property that can lead to useful features. For example, homomorphism makes it infeasible for block producers to create erroneous erasure codes (provided commitments are done correctly). This, along with other design decisions, allows Avail to avoid the need for fraud proof for block headers, a key differentiator vs. Celestia. Another example, homomorphism enables batch proofs, where a single proof can be used for multiple elements.
- KZG also integrates better with zk-proof systems. In fact, many zk-proof schemes utilize KZG.

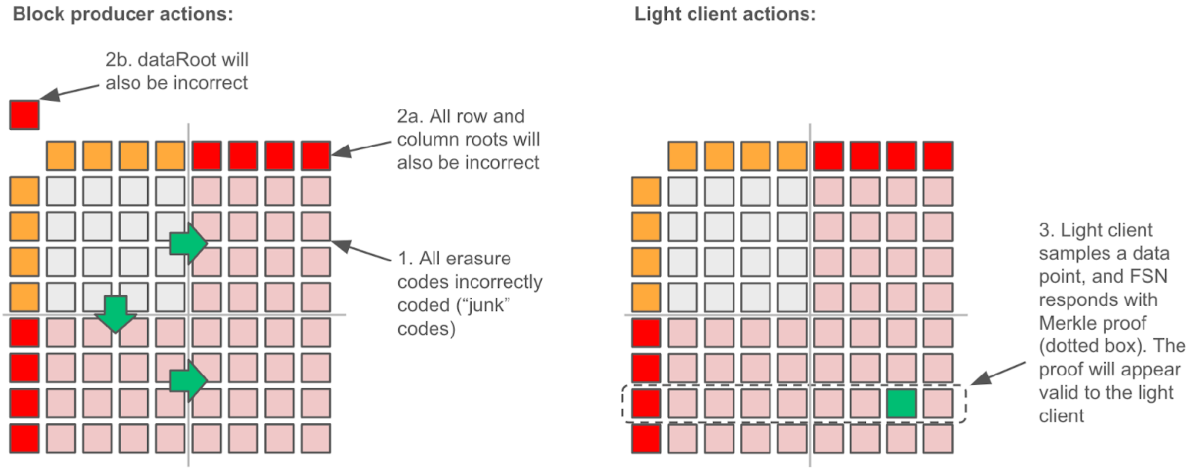
Disadvantages:

- KZG is more computationally intensive. Although both KZG and Merkle trees have constant commitment sizes, many implementations of KZG result in larger commitment sizes in absolute terms.
- Requires a trusted setup.
- KZG is not quantum resistant because it relies on elliptic curve cryptography. Merkle trees are believed to be quantum resistant.

We won’t dive into the mechanics of KZG, since that might be beyond the depth we are aiming for in this specific report. If you are interested, you can read the KZG paper(7) or Dankrad Feist’s blog post on KZG(8). Instead, we will **elaborate on one of the points we made earlier about KZG’s advantage of being partially homomorphic.**

Let's see how KZG thwarts a specific attack. In a 2D Reed Solomon (*a la* Celestia), compromised consensus nodes can produce incorrect erasure codes. For simplicity, let's assume validators produce a block where erasure codes are "junk" data, but all commitments are computed correctly over that "junk" data. This is a sensible scenario because computing erasure codes is more expensive than hashing.

LCs use these Merkle roots for data availability sampling. Notice that the system does not work if the Merkle roots are wrong in the first place.


Block producer actions:
2b. dataRoot will also be incorrect
2a. All row and column roots will also be incorrect
1. All erasure codes incorrectly coded ("junk" codes)
Light client actions:
3. Light client samples a data point, and FSN responds with Merkle proof (dotted box). The proof will appear valid to the light client

DA is no longer guaranteed, yet light nodes will not detect this with DAS. As far as light nodes are concerned, all data points exist and are correct.

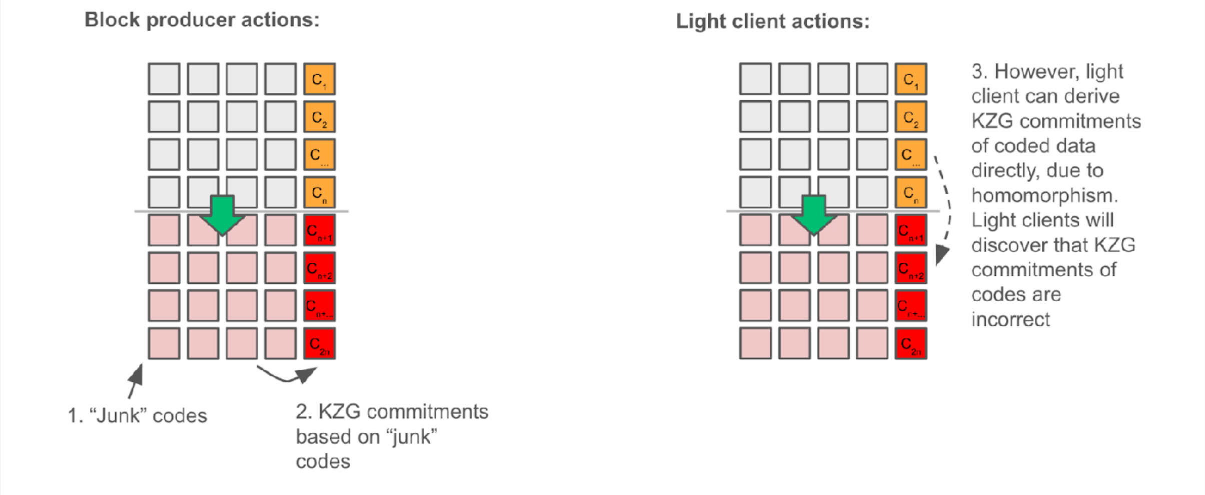
To counter this, Celestia has a fraud-proof mechanism where full nodes can challenge a block header. This allows Celestia to have security guarantees against certain attacks that are above the security of the Celestia chain. The downside of this "optimistic" approach is that LCs need to wait for a challenge period before performing DAS.

With KZG, this particular attack will not work. With KZG, LCs can check the commitments of the extended data without downloading the full dataset. Fraudulent validators may attempt two things, but either can be detected by LCs:

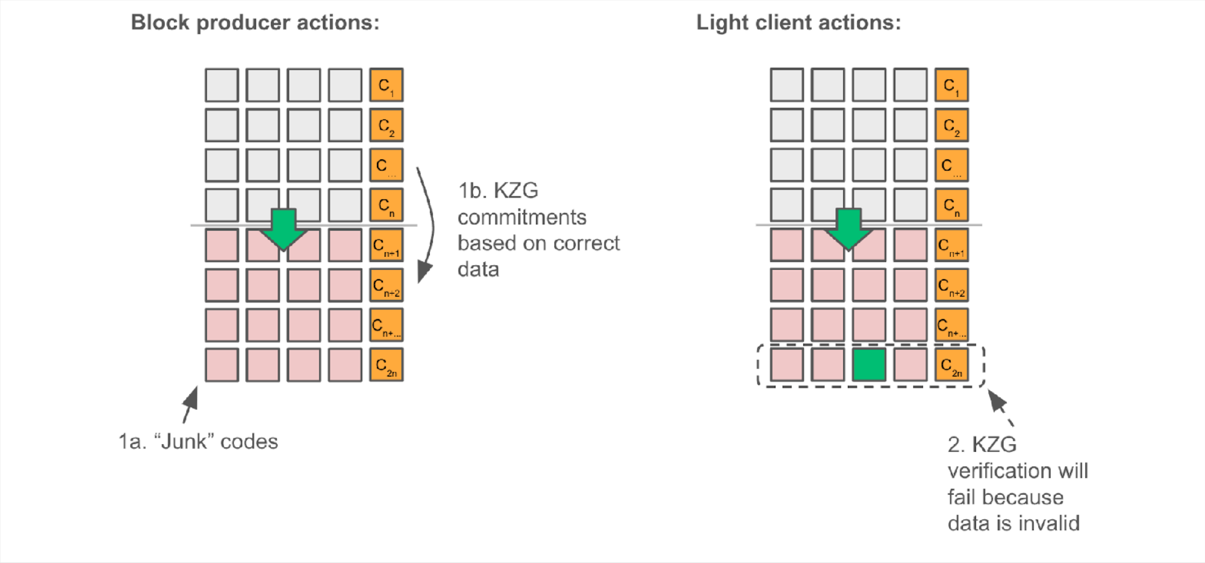
- Compute the commitments of the junk data, as before. This won't work because LCs can directly verify the correctness of these commitments.
- Compute the correct values for commitments Cn+1...C2n, but keep the junk data. This won't work either because now DAS will fail on any "junk" data, unlike with 2DRS.

Technical Explanation

Suppose full nodes perform a similar attack by producing “junk” codes and commitments based on the junk data. As before, a proof will appear correct to the LC. However, the homomorphic nature of KZG means that LCs can check that the commitments on extended data Cn+1…C2n are correct using C1…Cn (the reason is beyond the scope of this report, but note that it is not possible with hash functions used in 2D Reed Solomon).


Block producer actions:
C_1
C_2
$C_{...}$
C_n
C_{n+1}
C_{n+2}
$C_{n+...}$
C_{2n}
1. “Junk” codes
2. KZG commitments based on “junk” codes
Light client actions:
3. However, light client can derive KZG commitments of coded data directly, due to homomorphism. Light clients will discover that KZG commitments of codes are incorrect

Suppose now that the full nodes still produce incorrect erasure codes. But instead, commitments for extended data are derived based on the original data commitments. In this case, any proof of inclusion of the extended data will be invalid.



The designs of Avail and Danksharding are still in progress. However, this illustrates how KZG has useful features that can thwart certain attacks on incorrect erasure coding and ultimately avoid the need for a challenge period.

Rather than requiring clients to check the commitments the way we described, Avail's (and probably Danksharding's) current design requires the block producer to produce a validity proof that the erasure codes are correct. This "pessimistic" approach means that LCs do not need to wait for a challenge period before DAS, enabling faster DA finality. This is one of Avail's key differentiators.

[Avail's] "pessimistic" approach means that LCs do not need to wait for a challenge period before DAS, enabling faster DA finality. This is one of Avail's key differentiators.

KZG's useful features have other use cases. For example, it is used by EigenDA, but not in this manner. We will not elaborate on this in this report, for brevity's sake.

6 Outlook

Now that you have some idea of what DA is all about and how these different projects are tackling the issue, we can think about some potential effects this may have on the crypto markets in the coming weeks and months.

1. **Dedicated DA layers as R&D for full Danksharding?**
   - We can somewhat think of dedicated DA layers as conducting part of the research and development ("R&D") for full Danksharding; **helpful code and lessons learnt from these dedicated DAs will be able to be implemented into Danksharding by Ethereum.**

   - Or **is there an alternative universe where Ethereum stops developing Danksharding and focuses its attention elsewhere?** Remember, full Danksharding is a multi-year endeavor. If the likes of Celestia, EigenDA, Avail, and others can attain good market share and efficiency in this market, it might not make much sense for Ethereum to continue towards full Danksharding, at least not as a primary focus.

   - The interesting thing is that they all take different approaches (EigenDA, Avail, and Celestia) and make trade-offs.
     - **Different latency vs. bandwidth trade-offs,** e.g., how much communication must happen vs. how hard it is to re-construct blocks

    vs. whether you use a commitment scheme or not

    - It will be important to see which DA layer users prefer the (don't forget, the users of dedicated DA layers are rollup developers).

- In the most likely scenario that Ethereum continues to work on full Danksharding, there will be a long list of lessons that it can continuously take from these projects in the next years of development.

2. **New types of dApps**
    - It's less about cheaper rollups, but more about newer use cases and new types of dApps that developers can create with cheaper fees and a new DA model. **Projects are more likely to attempt high-throughput dApps (e.g., DePIN, AI) on L2s.** From a developer perspective, **EIP-4844 and dedicated DA layers have the potential to build new business models around fees** and the sequencer.

    - That said, the competitiveness of Ethereum L2s is increasing, likely **opening the gates for even more L2s than we have already seen.** Many were waiting for the first few to try out Celestia, while others were waiting for EIP-4844. Despite the positive impact of EIP-4844 so far, fees are likely to increase and somewhat normalize in the coming weeks and months as more L2s launch and transaction volume rises.

    - The **real moat of a DA layer is arguably the ecosystem that gets built using it and around it** - this is what ultimately drives the activity and traction of a DA layer. It will be important to monitor which project is able to attract the most projects to use it and further its ecosystem.

3. **Better user experience**
    - Affordable fees will encourage users to interact with more dApps and increase on-chain activity.

    - Can L2 developers **fully subsidize activity for users**? Maybe with EIP-4844 you can do that or be close to it if you're a dApp with a profitable model.
        - The concept of getting rid of gas or providing a **gas subsidy** could lead to a whole new wave of Web3 dApps.

        - Remember, every chain is now cheap and more competitive, so the "I have lower fees" argument is less of a differentiator compared to before.

4. **With DA getting closer to being optimized, does execution become the bottleneck to true scalability?**

5. **What is next for DA layers?**

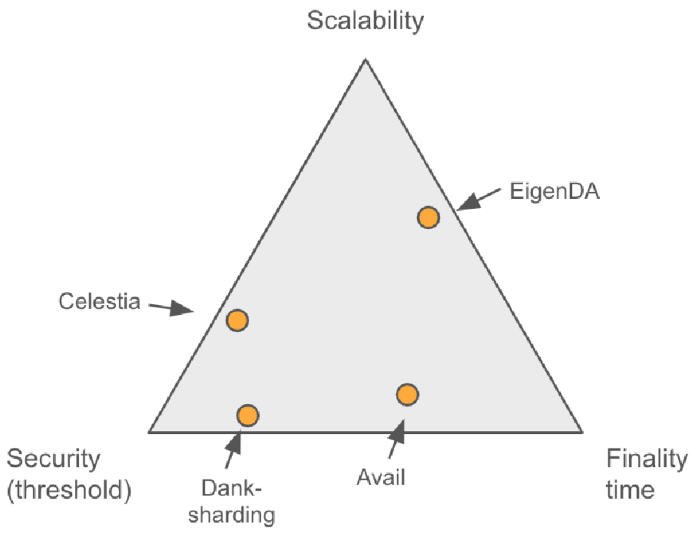
   - Value accrual is minimal as of now. There probably has to be some level of **vertical integration** in the future if DA layers are to become more profitable.

   - Is there a further question of **DA layers becoming commoditized** i.e., largely identical and offering a similar service? Does this lead to a further race to zero for fees?

As you can see, we are very much at the beginning of the DA conversation, with many problems worth considering and questions to be answered in the next few years.

7 Closing Thoughts

The DA protocol designs we explored uncovered a trilemma between scalability, security, and finality.



The placement of individual projects here is for illustrative purposes only, as a single dot is not a good way to represent design tradeoffs. It is based on current design and public information, which are fast-changing.

Celestia's and Avail's use of DAS brings high security (1- or 0-of-n honest full node requirement). Avail's use of KZG improves finality time, but at the cost of scalability to some extent. EigenDA relies on a dispersal protocol, which makes it more scalable but at the cost of security (honest supermajority requirement). All these still rely on the crypto-economic security of staked tokens to ultimately maintain the integrity of the protocol, whether it is native tokens or restaked ETH. Danksharding does away with this by directly integrating with Ethereum L1 to inherit full crypto-economic security. However, the tradeoff is lower scalability, as Ethereum will not be optimized for DA.

Different design decisions put protocols at an advantage on certain factors (like the use of KZG creating better time to finality). While true at a fundamental level, their implementations and optimizations matter a great deal too. We would likely see KPI comparisons that constantly shift as these protocols continue to mature. For example, just because EigenDA uses a dispersal protocol does not mean it will always have higher throughput, if another protocol does a better job at optimizing itself.

Also, it is yet to be seen which factor users care most about — is it cost, security, time to finality, etc.? Some projects may not care about the differences, and other factors such as proximity, familiarity, adoption, integrations, tooling, etc. may matter more. If so, this may put Danksharding at an advantage, as it can directly tap into Ethereum's large adoption.

There is also the existential question about what DA layers are really enabling. Is it really about scalability? Note that scalability, despite often being touted as the biggest problem in blockchain, has really only been a problem for the top couple of protocols. Or is it about creating a new landscape that is more modular and potentially more decentralized? Or is it about making it easier to bootstrap new chains (possible by combining a DA layer with a settlement layer)?

These remain open questions, which may be answered as the sector develops over the next few years.

References

1. https://ethereum.org/en/developers/docs/data-availability/
2. https://mirror.xyz/1kx.eth/gEzEkg_XP6arUQrvGp2cxjZ86vVMTmaXFdDiJax-z4I
3. https://arxiv.org/abs/1809.09044
4. https://arxiv.org/abs/2011.00102
5. https://github.com/availproject/data-availability/blob/master/reference%20document/Data%20Availability%20-%20Reference%20Document.pdf
6. https://arxiv.org/abs/2011.00102
7. https://www.iacr.org/archive/asiacrypt2010/6477178/6477178.pdf
8. https://dankradfeist.de/ethereum/2020/06/16/kate-polynomial-commitments.html

Latest Binance Research Reports

BINANCE RESEARCH

Monthly Market Insights

April 2024

Monthly Market Insights - April 2024

A summary of the most important market developments, interesting charts, and upcoming events

BINANCE RESEARCH

Navigating Crypto: Industry Map

March 2024

Navigating Crypto: Industry Map

An overview of different verticals in crypto

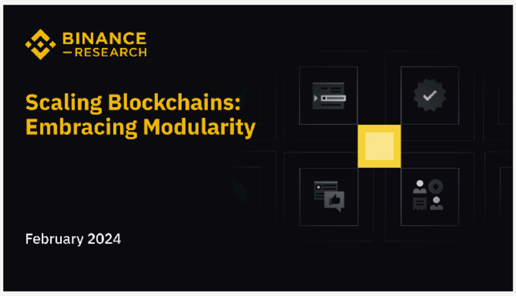
BINANCE RESEARCH

Orbiting the Cosmos Ecosystem

March 2024

Orbiting the Cosmos Ecosystem

An analysis of the Cosmos ecosystem



BINANCE RESEARCH

Scaling Blockchains: Embracing Modularity

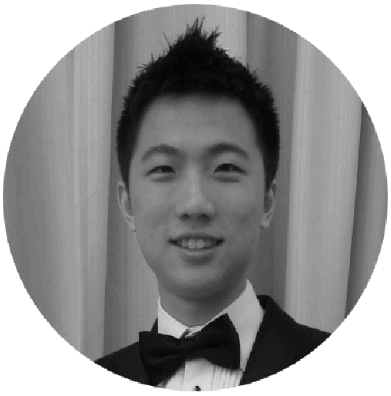
February 2024

Scaling Blockchains: Embracing Modularity

A look into the modular scaling landscape

About Binance Research

Binance Research is the research arm of Binance, the world's leading cryptocurrency exchange. The team is committed to delivering objective, independent, and comprehensive analysis and aims to be the thought leader in the crypto space. Our analysts publish insightful thought pieces regularly on topics related (but not limited) to the crypto ecosystem, blockchain technologies, and the latest market themes.



Derek Ho

Protocol Specialist

Derek is a Protocol Specialist at Binance, where he analyzes protocols for their technical design, feasibility, performance, and security. He works with various teams, aiding in understanding and assessing different technologies. He enjoys reading whitepapers, mathematical formulas, and blockchain code. He holds an engineering degree from Cambridge University.

Shivam Sharma

Macro Researcher

Shivam is currently working for Binance as a macro researcher. Prior to joining Binance, he worked as an investment banking associate and analyst at Bank of America on the Debt Capital Markets desk, specializing in European financial institutions. Shivam holds a BSc in Economics degree from the London School of Economics & Political Science ("LSE") and has been involved in the cryptocurrency space since 2017. Follow him on X: @**Sh_ivam**.

Resources

Read more here

Share your feedback here

General Disclosure: This material is prepared by Binance Research and is not intended to be relied upon as a forecast or investment advice and is not a recommendation, offer, or solicitation to buy or sell any securities or cryptocurrencies or to adopt any investment strategy. The use of terminology and the views expressed are intended to promote understanding and the responsible development of the sector and should not be interpreted as definitive legal views or those of Binance. The opinions expressed are as of the date shown above and are the opinions of the writer; they may change as subsequent conditions vary. The information and opinions contained in this material are derived from proprietary and non-proprietary sources deemed by Binance Research to be reliable, are not necessarily all-inclusive, and are not guaranteed as to accuracy. As such, no warranty of accuracy or reliability is given, and no responsibility arising in any other way for errors and omissions (including responsibility to any person by reason of negligence) is accepted by Binance. This material may contain ’forward-looking’ information that is not purely historical in nature. Such information may include, among other things, projections and forecasts. There is no guarantee that any forecasts made will come to pass. Reliance upon information in this material is at the sole discretion of the reader. This material is intended for information purposes only and does not constitute investment advice or an offer or solicitation to purchase or sell any securities, cryptocurrencies, or any investment strategy, nor shall any securities or cryptocurrency be offered or sold to any person in any jurisdiction in which an offer, solicitation, purchase or sale would be unlawful under the laws of such jurisdiction. Investment involves risks.