This post was co-authored by @STB and @nashqueue

This post is by no means intended to be a comprehensive resource for fully understanding the DA sampling of Celestia or ZODA in a rigorous way. Rather, it aims to sketch a high-level idea of what ZODA can potentially enable in the current scheme of Celestia DA Sampling. For a complete understanding of either of these papers, readers are strongly encouraged to refer to the original publications.

Introduction

Background

One core value of the Celestia community is that blockchains must be directly verifiable by all to meaningfully replace centralized systems. Reflecting this principle, the community has prioritized the development and adoption of trust-minimized light nodes, enabling users to directly verify the correctness of the chain using techniques such as data availability sampling. This technique provides light clients with assurance that the block data needed for full nodes to generate fraud proofs is available, as long as there are a sufficient number of honest light clients to reconstruct missing data from blocks. The more light clients participating, the higher the reconstruction probability, which benefits the entire Celestia network.

The effort towards enabling more light clients is exemplified by recent developments, such as enabling end-users to run a light node while accessing Celenium, a blockchain explorer for Celestia. Another example is the Leap Wallet, which integrates a light node into the wallet. This not only increases the number of data samples but also reduces reliance on centralized endpoints, allowing users to independently verify the data displayed in both the explorer and wallet, further reinforcing trust minimization.

Minimizing trust is another critical aspect for light clients, that is enabling them to detect invalid blocks independently and without relying on trust committees, including Celestia's consensus nodes. For that reason, Celestia initially implemented fraud proofs, which allow light nodes to verify block validity even in the presence of an honest majority attack of the consensus nodes. In this design, at a high level, light nodes consider a block as valid if during the block's fraud proof window, no fraud proof is released by full nodes (assuming that each light client is connected to at least one honest full node).

Techniques that aim at minimizing trust, including fraud proofs, present an important challenge which is providing fast finality for light clients. That is reducing the time it takes for light clients to determine whether a block is valid or invalid. In fraud proofs, light clients finality is subject to delay due to the fraud proof window.

With advancements in zero-knowledge (ZK) technology, Celestia can revisit its reliance on fraud proofs, exploring alternative mechanisms that further minimize the trust assumptions of light clients, enhance time required for finality, while enabling faster user experience for light nodes. One such alternative mechanism, though not ZK-based (in the sense that there is no "zero-knowledge" property guarantee), that has been considered to enable fast finality is ZODA, Zero-Overhead Data Availability, which is the focus of this research post.

Celestia DA Network and Light Clients Finality

In the Celestia Data Availability network, light nodes perform random sampling of the Reed-Solomon (RS) encoded block data, where the block data is arranged into a square of fixed-size byte chunks called shares, and the sampling occurs at the share level. Upon verifying these sampled shares, light nodes make them available to full nodes. With enough samples, a full node can reconstruct the block and verify its validity, ensuring that the erasure coding is correct and adhering to other validity rules—or otherwise generating fraud proofs if inconsistencies are found. Light nodes will consider a block available once the fraud proof window for that block has passed without any fraud proofs issued by a full node. This means block finality from a light client's perspective is delayed by the fraud proof window. However, if light client finality could become faster (without trusting the honest majority of the consensus layer)—where each light client could independently ensure the encoding's correctness without waiting for a fraud proof from a full node—it would enhance network scalability and align with trust minimization, one of the DA network's core objectives.

Why ZODA

In the quest for faster finality for light clients, this research post explores integrating the Zero-Overhead Data Availability (ZODA) scheme into Celestia. ZODA, at a high level, is an innovative approach that allows anyone to sample a small number of rows and columns from the tensor-encoded matrix and verify that, with high probability, these samples are correctly encoded versions of (more specifically, are "close" to a correct encoding of) some data X without needing to download the entire matrix. This is achieved through a smart and small modification to the encoding process. This protocol also avoids any reliance on trusted setup, aligning further well with the goal of trust minimization. Building on this concept, we examined whether ZODA's encoding scheme could be incorporated into Celestia's square construction, allowing light client samples to directly serve as proofs of correct encoding, hence eliminating the need for fraud proofs for incorrect encoding and enabling faster finality for light clients.

In the initial phase of this research effort, summarized in this post, we would like to shed light on the current state of efforts regarding the assessment of the computational cost associated with the ZODA-based square construction as well as present research findings and discussion points related to removing fraud proofs, and achieving faster finality in the Celestia DA network with the integration of this new encoding scheme.

What is ZODA?

The ZODA protocol comprises three algorithms: Encoding, Sampling, and Decoding, as explained below in general terms. The components related to the discussions of this research post are Encoding and Sampling.

- Encoding: This algorithm encodes input data, represented as a matrix, into a form where each row and column serves as a proof of correct encoding. It does so by incorporating randomness derived from a partially encoded matrix and randomizing part of the matrix before completing the tensor encoding.
- Sampling: This algorithm enables verification of the encoding by sampling a small number of rows and columns. It verifies that the purported encoding is ("close to") an encoding of some unique message, and that the sampled rows and columns correspond to the correct encoding of this unique message, with high probability. There is a mathematical definition of what it means for an encoding to be "close to" an encoding of some data or "far from" any valid encoding. However, for simplicity, we will skip the details and provide only the high-level idea. Sampling is communication-efficient, requiring no additional data or proofs beyond the sampled portions of the encoding.
- Decoding: Anyone who has run the sampling protocol and subsequently received enough rows and columns of the encoding can verify that these rows and columns are correctly encoded. If sufficient checks pass, they can then recover the unique encoded message.

ZODA is considered zero-overhead because the resulting encoding requires no additional storage and only incurs a minimal increase in computational effort compared to standard tensor encoding.

Integrating ZODA into Celetia: High Level Design

When integrating ZODA into the Celestia Data Availability network, validators will utilize the ZODA encoding algorithm to generate the extended data square (an overview of the encoding process is provided in the next section). Light nodes will employ the ZODA sampling algorithm to verify the correctness of this encoding. In the current design of Celestia, the correctness of the encoding is verified by full nodes—not light nodes—by re-encoding the entire data square and checking that the encoding is done correctly. It turns out that this process requires full nodes to perform redundant encoding.

Celestia Extended Square Construction using ZODA at High Level

A high-level overview of the ZODA-based extended square construction is provided below.

Let the extended data square be:

Where Q_i's represent quadrants, each of size $n \times n$.

For this specification, we assume each quadrant consists of field elements rather than shares. In Celestia square construction, shares are 512-byte data chunks, differing from elements in fields like $GF(2^8)$ or $GF(2^{16})$. Therefore, each quadrant consists of $n \times n$ field elements.

Q1 Computation

Q1 is the original data square.

Q3 Computation

Encode Q_1 column-wise using Reed-Solomon encoding. Place the parity shares into Q_3.

Q2 Computation

Let X be the leftmost side of the extended data square consisting of Q_1 and Q_3, i.e.:

1. Commit to each row of X as follows:
2. Compute the hash of all concatenated elements in each row. Alternatively, each row can be Merkleized, though this is not essential for ZODA construction.
3. Construct a binary Merkle tree where each row's hash (from the previous step) forms a leaf, resulting in a tree with 2n

leaves, where n is the width of each quadrant.

4. Let c denote the root of this Merkle tree.
5. Calculate D_r, a diagonal matrix of dimension n * n with random 128-bit elements from GF(2^{128}) along the diagonal. Using Fiat-Shamir technique, one way to compute D_r is as follows, with diagonal elements shown:

[

image

722×373 4.71 KB

](https://forum.celestia.org/uploads/default/original/2X/8/8085a1b8efbc79c10016f78c1f0b7bf993ee8ff6.png)

1. Let $Q_{1r} = Q_1 \cdot D_r$. At this stage, a field extension is required since the fields used for RS encoding of Q_1 and Q_3 are GF(2^8) (or GF(2^{16})) whereas the elements in D_r are from a larger field, i.e., GF(2^{128}). Consequently, the multiplication and RS encoding (steps 3 and 4 below) occur within this larger field. This creates an asymmetry between the left and right sides of the extended data square. One solution is to perform RS encoding in the larger field GF(2^{128}) for all quadrants from the beginning.
2. Encode every row of Q_{1r} and place the parity shares in Q_2.

Some implementation details about the calculation of D_r: Assuming each element of diagonal matrix D_r is from GF(2^{128}) and SHA-256 is used as the hash function for D_r calculation, each 256-bit hash output could be split into two elements (i.e. as two 128-bit values). Therefore, only n/2 hashes would be needed to generate the elements on the diagonal of D_r. However, the exact instantiation of D_r and its security implications are yet to be finalized and explored in the implementation.

Q4 Computation

Encode every column of Q_2 and place the parity shares into Q_4.

Final Commitment

Compute a Merkle commitment over the Merkle commitments of the columns of the final extended square.

Alternative Ways of Computing Q2 and Q4

Once Q_1 and Q_3 (i.e., X) are calculated, multiply D_r with both Q_1 and Q_3, then:

- Encode $Q_1 \cdot D_r$ row-wise and place the parity shares in Q_2.
- Encode $Q_3 \cdot D_r$ row-wise and place the parity shares in Q_4.

Prototyping the construction

The initial prototyping aimed to evaluate the computational overhead that the new ZODA square construction might impose on validators and to compare it with the current Celestia square construction.

The benchmarking did not include the following aspects:

- Sampling
- Reconstruction
- Proof Verification
- Running on a blockchain

The core logic for square construction resides in the rsmt2d library, which has been extended with the ZODA implementation specifically for benchmarking purposes.

Discussion points, findings, and challenges:

We have started prototyping the ZODA-based extended square construction and conducting benchmark tests in Go. The majority of the benchmarking is done. However, computing the right side of the square (i.e., Q2 and Q4) presented a technical challenge due to the requirement of using a larger field, as mandated by ZODA. While the field used for RS encoding of Q1 and Q3 is GF(2^8) (or GF(2^{16})), the elements in D_r belong to a larger field, i.e., GF(2^{128}). Consequently, both the multiplication and RS encoding during the computation of Q2 and Q4 must occur in this larger field. The challenge was

that the current library used in rsmt2d does not support RS encoding in GF(2^{128}). The completion of the benchmarking requires the following next steps:

1. Exploring existing Reed-Solomon libraries that support GF(2^{128}) and porting them into Go to complete the benchmarking. One such library in Rust has already been identified and if deemed feasible, could be ported to Go (alternatively, continuing the benchmarking in Rust could be considered).

2. One potential performance challenge in this new construction lies in encoding the second half of the square in the larger field. If this is determined to be a limiting factor, based on the benchmarking results, the following actions could be considered:

3. Developing a more efficient, hardware-accelerated library that supports GF(2^{128}).

4. Investigating a theoretical approach to assess whether RS encoding the right half of the square in GF(2^8) (or GF(2^{16})) could still achieve the same guarantees provided by ZODA with GF(2^{128}).

Integrating ZODA into Celetia: Findings and Discussion

Agreement and Finality

In the current design of Celestia, fraud proofs from full nodes providing evidence of bad encoding are to enable the agreement property. According to the agreement property (simplified for the forum post): If an honest light client accepts a block as available, then all other honest light clients will also accept that block as available within some known maximum delay. Therefore, when assessing whether ZODA could enable the removal of fraud proofs, it is essential to also consider whether the agreement property holds and under what conditions it is maintained.

For agreement to be preserved, we need to show that an adversarial encoder/prover cannot partition sampling light nodes into two groups: one accepting that the square is correctly encoded and one rejecting it as incorrectly encoded. When sampling in ZODA, there are three cases:

1. If a square is correctly encoded (i.e., has no errors), light clients sampling it will successfully verify it.

2. If a square is far from any valid encoding, the ZODA sampling algorithm will detect this, except with a small soundness error probability.

3. In a third scenario, a square may be close to a valid encoding of some data but not fully correctly encoded, meaning it may contain some errors. For such a square, some light client samplings may succeed, while others fail (verification fails), resulting in no immediate agreement among light nodes. Due to this, a secondary mechanism is required to inform those failed light nodes on whether the encoding is done correctly or not.

Below, we sketch different potential ways of addressing finality in this construction, based on the above enumerated situations.

When the sampling is verified successfully, i.e., case 1 and part of 3, it means that the encoding is done good enough (Close to a correct encoding) to the extent that the underlying square can be reconstructed using the correctly encoded parts. In that case, the node can safely move on and consider the encoding correct. However, the issue about finality arises when the sampling verification fails, which can occur in case 2 or 3. In case of 2, the node can reject the encoding. A failed verification in the third case can be seen as a false negative. This occurs when the light node samples a portion of the encoded square containing errors, even though enough of the square is correctly encoded to allow reconstruction using the valid portions. Therefore, a verification failure does not mean the square is not reconstructable. In such a case, there needs to be a way to inform and unlock the failed light nodes. A possible approach could be that light nodes whose verification has passed use their correct shares to reconstruct the correct versions of the faulty or badly encoded shares and provide these (along with proof that the reconstruction was performed correctly) to the failed nodes. This allows the failed nodes to verify the correct shares and get unblocked. This approach is more involved and may come with some trade-offs that require further research.

In terms of finality, if ZODA verification passes, light nodes can consider the block decodable and move on hence can benefit from a faster finality needless to fraud proofs. This means that as long as no honest majority attack happens in the consensus layer, squares are expected to always be correctly encoded, hence the network will enjoy faster finality. However, if ZODA verification fails (which is likely to be the case only when there is honest majority attack), light nodes are subject to some delay due to the secondary mechanism which is needed to unblock them and inform them about the state of the square encoding i.e., whether being close to a correct encoding or not.

Final Notes and Research Ideas

Building on the discussions in the current research post, integrating ZODA into Celestia can facilitate faster finality when ZODA sampling verification succeeds, particularly in the absence of an honest majority attack in the consensus layer. This contrasts with the current design of Celestia, where the fraud-proof window must always be observed. However, in cases

where ZODA sampling verification fails, a secondary mechanism is needed to help the failed light nodes determine whether the failure is a false negative or not. This secondary mechanism may introduce delays, which could hinder the goal of achieving faster finality. Trade-offs around different approaches to this secondary mechanism require further exploration and research.

In the third case discussed in the previous section i.e., “Agreement and Finality,” where a square is close to a valid encoding of some data but not fully correctly encoded, several potential research directions arise:

- Ensuring Commitment Validity: Investigating the implications of ZODA commitments being computed over data that may contain encoding errors rather than an exact encoding. A key area of exploration could involve developing methods to ensure that inclusion proofs to the commitments can only be generated for correct shares, preventing the possibility of proving the existence of a blob in the square that does not exist in the correctly constructed square.
- Establishing Feedback Loops With Consensus Layer: Exploring whether commitments over partially encoded squares should be recomputed—and potentially re-signed by consensus nodes—to reflect the correct encoding. If so, the specific processes and protocols required for this adjustment must be explored.

As a solution for both agreement and reducing light node download size at the expense of some prover overhead, we can construct a recursive proof of ZODA. Verifying this proof may allow light nodes to perform distributed decoding to regain agreement even in the presence of errors. We are actively exploring this approach. More on this soon.

Acknowledgments

We would like to thank Alex Evans and Guillermo Angeris for their invaluable help in this research effort and for deepening our understanding of ZODA and its application for Celestia. We would also like to thank @musalbas and @adlerjohn for the continued discussions to help us understand the nuances of data availability.

Resources

- ZODA: Zero-Overhead Data Availability
- ZODA: An Explainer
- Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities