The Accidental Computer: Polynomial Commitments from Data Availability

Alex Evans
aevans@baincapital.com

Guillermo Angeris
gangeris@baincapital.com

January 2025

Abstract

In this paper, we show two simple variations of a data availability scheme which enable it to act as a multilinear polynomial commitment scheme over the data in a block. The first variation enables commitments over all of the block's data with zero prover overhead: the data availability construction simply serves both purposes. The second variation allows commitments over subsets of data with nonzero but still concretely small proving costs, since most work is already done during data encoding. This works especially well for blockchains with a high degree of data parallelism, as data-parallel computation is particularly amenable to efficient GKR proofs. Since, in GKR, opening the polynomial commitment contributes significantly to prover costs, our construction enables the prover to reuse work already done by the data availability scheme, reducing—or wholly removing—work associated with the polynomial commitment scheme.

1 Introduction

Blockchains, as their name suggests, are composed of an ordered list of blocks. These blocks may include the data associated with financial transactions, payments, or even social media posts and interactions. An important part of this construction is that, as new blocks are added to this list, these new blocks must satisfy some computational rules (or predicates) which ensure that the data in the blocks is 'reasonable'. For example, one rule (of potentially many) may be that no transaction in the block may transfer more money out of an account than the account has available; *i.e.*, that account balances may not be negative. The list of such predicates is often called the *state transition function* of a blockchain: the predicates ensure that the latest state of the blockchain, once it has been modified by the transactions of a block, is still a valid end state.

Designs. In early blockchain designs, verifying the state transition function required downloading all transaction data for each block and verifying the predicates for all downloaded

blocks. As blockchains scale to support more concurrent users and therefore bigger blocks with more transactions, resource-constrained nodes can no longer participate in this validation. As a result, blockchains faced a hard trade-off between scalability and decentralization. A proposed solution to this problem is to enable so-called verifying light clients [ASB18]: resource-constrained clients that can verify the validity of the chain without needing to download all blocks and verify the validity of each. To this aim, a light client must ensure that (a) the state transition function correctly verifies without re-executing it on all blocks, while also (b) verifying that the transaction data in the blocks is accessible and unique, ideally without needing to download all transaction data.

Succinct proofs. To partially solve this problem, blockchains have turned to *succinct proofs* [Tha22, EA23]—short proofs which enable the verification of some statement in much less time than is required to evaluate the statement in the first place. Succinct proofs solve problem (a) essentially by construction: light clients wish to verify some predicate on data and receive a succinct proof that this predicate indeed evaluates to 'true'. Since this proof is often much easier to verify than actually computing the predicate on the data, light nodes can easily perform the verification work of (a) even on very small machines such as phones or laptops.

Data availability. This leaves the verification work of (b); *i.e.*, ensuring that the data over which the succinct proof is constructed is available to download. If the blocks are of small size, this is relatively simple: the light node just downloads the whole block and verifies that the predicate passes. As blocks scale, this becomes more difficult on devices with limited bandwidth. To this end, data availability protocols such as [ASB18,HSW23,HSW24,EMA24] have been constructed. These allow light nodes to download only small parts of the latest block to ensure that the data is available.

Simple observations. One simple observation is that a significant portion of the work done in the construction of many succinct proofs is in encoding some statement (using a linear code, for example). This is also the same work done in the encoding step of many data availability schemes. A natural question then, is: can we re-use the encoding and/or sampling work done in the data availability scheme to reduce the work required to prove or verify a succinct proof over this data?

This work. In this work, we show that it is possible to enable two types of state transition functions to be succinctly verified for a blockchain running a data availability scheme such as the ZODA tensor variation [EMA24, App. E]. In particular, any blockchain which runs the encoding and sampling steps described above can reuse much (if not all) of the work done in the encoding and sampling steps to efficiently prove a statement over the data of the latest block. This can, in turn, be verified by the light clients without requiring the whole block be downloaded. We also suspect this observation is widely applicable to a number of

other data availability schemes such as suitable adaptations of FRIDA [HSW24] that support multilinear evaluation, but focus on this special case in order to give a practical construction.

2 Tensor ZODA variation

In this section, we give a brief introduction to the tensor ZODA variation of [EMA24, App. E], along with its guarantees, which we use in what follows. In this section, and in the rest of the note, we will have two (linear) codes $G \in \mathbf{F}^{m \times n}$ and $G' \in \mathbf{F}^{m' \times n'}$ over some finite field $\mathbf{F}$, each with distance d. (The protocol is flexible enough to allow for different distances, but we assume this for convenience, leaving the relatively simple extension for later.) We denote $\tilde{X} \in \mathbf{F}^{n \times n'}$ as the data for which we wish to both encode and prove correctness. We set $\|X\|$ to be the number of nonzero rows of a matrix $X \in \mathbf{F}^{m \times m'}$ and interpret any n-vectors as $n \times 1$ matrices, when convenient, such that $\|x\|$ is the number of nonzero entries of the n-vector $x \in \mathbf{F}^n$. For a matrix X, given a subset of indices $S \subseteq \{1, \ldots, m\}$, we write X_S for the submatrix containing the S rows of X (in any standard order). Finally, we define

$$\bar{g}_r = (1 - r_1, r_1) \otimes \cdots \otimes (1 - r_k, r_k), \tag{1}$$

where $r_1, \ldots, r_k \in \mathbf{F}$ are uniformly randomly drawn and $\otimes$ denotes the Kronecker product such that $\bar{g}_r \in \mathbf{F}^{2^k}$ and similarly for $\bar{g}'_{r'} \in \mathbf{F}^{2^{k'}}$. This type of 'structured randomness' (sometimes called logarithmic randomness [DP24b, AER24]) will allow us later to efficiently evaluate a polynomial whose coefficients are in $\tilde{X}$.

2.1 Tensor ZODA variation

The two relevant parts of the tensor ZODA protocol we use in this paper are those of the encoder and those of the sampler. We will see in what follows that the encoder is essentially performing most of the work needed to prove that a particular row or collection of rows from $\tilde{X}$, when interpreted as the coefficients of some multilinear polynomial, evaluates to some stated value. As a side effect, we will also see that almost all of the work needed to show that the entirety of $\tilde{X}$, when viewed as the coefficients of a multilinear polynomial, evaluates to some value, queried at a uniformly random point. These polynomial evaluations and corresponding proofs usually make up a significant proportion of the proving time in a succinct proof system. Since the work to construct these proofs already needs to be done to prove the correctness of the encoding and is already needed for data availability sampling, this results in potentially substantial net savings in both computation and communication for the sampler node to verify proofs over the committed data $\tilde{X}$.

2.2 Encoder algorithm

The work of the encoder algorithm is relatively simple. It begins with the data matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$ and is described below.

1. Encode $\tilde{X}$ to get $Z = G\tilde{X}G'^T$
2. Commit to the rows of Z and, separately, to the columns of Z
3. Receive some uniform randomness $r \in \mathbf{F}^k, r' \in \mathbf{F}^{k'}$ and construct $\bar{g}_r \in \mathbf{F}^{n'}$ and $\bar{g}'_{r'} \in \mathbf{F}^n$
4. Compute $y_r = \tilde{X}\bar{g}_r$ and $w_{r'} = \tilde{X}^T\bar{g}'_{r'}$
5. Send y_r and $w_{r'}$

We note that the randomness here is public coin and can therefore be either sent by the sampler or, in more practical settings, derived using the Fiat–Shamir heuristic from the commitments to the rows and columns of Z. Additionally, though we do not explicitly do so here for compactness, the randomness may be drawn not just from the field $\mathbf{F}$, but, indeed, from any field extension $\mathbf{F}' \supseteq \mathbf{F}$ to decrease the error probability of a test erroneously passing. Finally, we have implicitly assumed that n and n' are powers of two in order to be compatible with the construction of $\bar{g}_r$ and $\bar{g}'_{r'}$ in (1), although it is easy to extend to the more general setting.

2.3 Sampler algorithm

The sampler algorithm is also relatively simple. Note that since the commitments to the rows and columns of Z are not a-priori known to be of the same matrix, we will call Y the matrix whose commitment should correspond to the rows of Z and W the matrix which should correspond to the columns. (The verifier will then convince itself that indeed Y and W are close to the rows and columns, respectively, of $G\tilde{X}G'^T$, the tensor encoding of some matrix $\tilde{X}$.) We assume that the randomly drawn elements $\bar{g}_r$ and $\bar{g}'_{r'}$ are publicly known and that y_r and $w_{r'}$ have been received by the sampler. All received rows and columns are also assumed to be checked against the commitments provided by the encoder.

1. Receive two commitments, one over the rows of Y and one over the columns of W
2. Uniformly sample $S \subseteq \{1, \ldots, m\}$ of (fixed) size $|S|$
3. Uniformly sample $S' \subseteq \{1, \ldots, m'\}$ of (fixed) size $|S'| = |S|$
4. Receive the S rows of Y and S' columns of W (*i.e.*, receive $(W^T)_{S'}$)
5. Verify the received rows of Y are codewords of G'; *i.e.*, that $Y_S = \bar{Y}_S G'^T$
6. Verify the received columns of W are codewords of G; *i.e.*, that $(W^T)_{S'} = \bar{W}_{S'} G^T$
7. Verify that $\bar{Y}_S\bar{g}_r = G_S y_r$
8. Verify that $(\bar{W}^T)_{S'}\bar{g}'_{r'} = G'_{S'}w_{r'}$
9. Verify that $\bar{g}'^T_{r'}y_r = w^T_{r'}\bar{g}_r$

If any verification step fails, the sampler running this algorithm does not accept. For a 'pictorial' representation of the committed matrices Y and W, and the vectors y_r and $w_{r'}$, see figure 1. In this figure, the dashed entries indicate that these are (systematic) encodings of the underlying vectors. Finally, the check in step 5 (and similarly for 6) can be interpreted as 'there exists a matrix $\bar{Y}_S$ whose row encodings match the S subsampled rows of Y'; that is, that each of the S subsampled rows of Y are indeed encodings of some messages, via G'^T.

High level interpretation. The idea here is that the sampler is, roughly speaking, receiving a structured random linear combination of the columns of the unencoded message (y_r) and verifying that this combination, after encoding, is consistent with a small number of randomly checked rows of Y. (The verifier also checks that each of these rows are a valid codeword, since Y is supposed to be a tensor code.) This implies that the matrix Y is (row-wise) close to some unique, tensor encoded matrix $G\tilde{Y}G'^T$. A similar thing is true for W and some matrix $G'\tilde{W}G^T$, except over its columns, instead of its rows. Finally, the last step simply checks that taking a structured random linear combination of the columns of the unencoded message (using $\bar{g}_r$) and then combining this result (using $\bar{g}'_{r'}$) is the same as taking a structured random linear combination of the rows of the unencoded message (using $\bar{g}'_{r'}$) and then combining the result (using $\bar{g}_r$).

Discussion. While verifying that the received rows and columns of Y and W are codewords and decoding to $\tilde{Y}$ and $\tilde{W}$ may seem like a computationally expensive task, we note that, in most cases, G and G' are systematic codes (*i.e.*, the message corresponding to the codeword is part of the codeword itself) and hence can be easily checked by encoding the appropriate entries of Y and W. We assume this for the rest of the paper for convenience, but note that it is not strictly necessary as the encoder can simply send the corresponding codewords $\tilde{Y}_S$ and $\tilde{W}_{S'}$ to the sampler who can verify the results against the respective committed rows and columns. Finally, there is no strong requirement that $|S| = |S'|$ (and this flexibility might be helpful if G and G' have different relative distances) but, for convenience, we assume that $|S| = |S'|$ and that $d = d'$ since we expect this will be the standard choice in practice.

2.4 Guarantees

While the sampling algorithm presented above [EMA24] provides a number of useful guarantees, we will use three important ones and present proofs of their correctness in appendix A.

Unique decoding. The first is that the sampler, upon a successful completion of the sampling algorithm, knows that there exists a unique matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$ which is closest to the committed matrix Z. More specifically, the rows of a tensor encoding of $\tilde{X}$ are close to the rows of Y and the columns of the tensor encoding are close to the columns of W within the unique decoding radius; *i.e.*,

$$\|Y - G\tilde{X}G'^T\| < d/3 \qquad \text{and} \qquad \|W^T - G'\tilde{X}^TG^T\| < d/3, \tag{2}$$

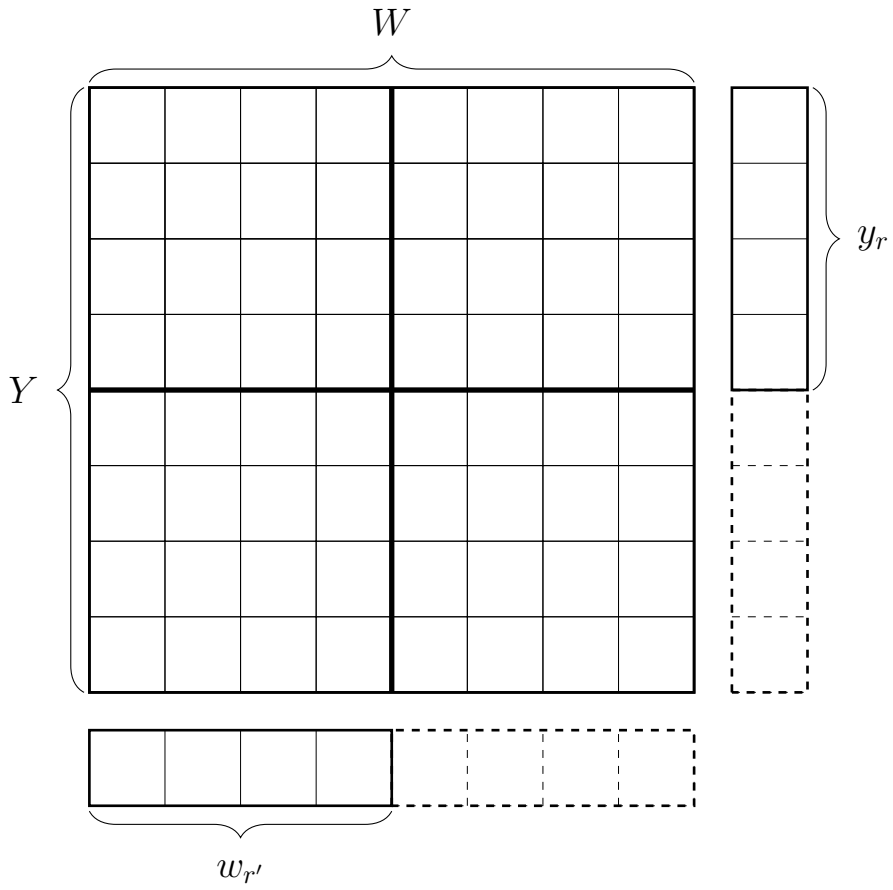

Figure 1: An illustration of the encoded square and the corresponding proofs used by the sampler to verify correctness.

except with error probability no more than

$$\left(1 - \frac{d}{3m}\right)^{|S|} + \frac{k(d+9)}{3|\mathbf{F}|}, \tag{3}$$

over the randomness $\bar{g}_r$, $\bar{g}'_{r'}$, S, and S'. (The constants can be slightly improved if G and G' are Reed–Solomon codes [DG24], but we do not do so here for simplicity.)

Matrix-vector product. The second, is that the received vector y_r satisfies

$$y_r = \tilde{X}\bar{g}_r, \tag{4}$$

where $\tilde{X}$ is the unique matrix of (2), except with error probability no more than the one given in (3). (A similar thing is true for $w_{r'}$ and $\bar{g}'_{r'}$ in that $w_{r'} = \tilde{X}^T\bar{g}'_{r'}$ except with probability no more than given in (3).) Note that equation (4) is an *exact equality* and will be used in what comes next.

Correct encoding. Finally, any sampler that has run the sampling algorithm will have, using the previously-derived matrix-vector product property, a guarantee that any of the S received rows of Y (or columns of W) correspond to correctly encoded rows of the unique $\tilde{X}$; *i.e.*:

$$Y_S = G_S\tilde{X}G'^T.$$

Except with error probability no more than that of (3). Additionally, any further received rows of Y, say $\tilde{S} \subseteq \{1, \ldots, m\}$ which also satisfy

$$Y_{\tilde{S}}\bar{g}_r = G_{\tilde{S}}y_r,$$

will also have the property that they are correctly encoded rows of $\tilde{X}$:

$$Y_{\tilde{S}} = G_{\tilde{S}}\tilde{X}G'^T,$$

again, except with error probability no more than (3).

Discussion. We will see how to use each of these properties in the next two sections, in two different settings, to significantly reduce the amount of proving work that has to be performed. In the first setting, we will prove a statement over the complete block. In the second setting, we will prove a statement over some (fixed) subset of the data in a provided block. The idea here is that during both the encoding and the sampling steps, the encoder (and sampler) have performed most of the work needed to open a polynomial over the underlying data at a given point. Since the encoding and sampling algorithms already need to be run to ensure the data is available, this substantially reduces the work necessary to prove a statement (or predicate) over the underlying data.

3 An aside on succinct proofs

We will briefly discuss a very high level view of how one might succinctly prove a general computation over some piece of data has been correctly performed. Readers very familiar with general purpose succinct proof systems such as GKR [GKR08, ZGK+17, Tha22] should feel free to skip this section and continue to the next.

GKR protocol. In a general computation, we would like to show that some function C, defined as some finite computation, when evaluated on the inputs $\tilde{X}$, has some output, say $C(\tilde{X}) = z$. It is not hard to show—though involved to make precise—that any (finite) computation may be represented as a sequence of additions and multiplications over a finite field. The GKR protocol [GKR08] (and its improvements [ZGK+17, XZZ+19]) are a way of succinctly reducing the verification of $C(\tilde{X}) = z$ (which may be constructed as a sequence of addition and multiplication gates) to verifying that $\tilde{X}$, when viewed as the coefficients of a multilinear polynomial, evaluates to a particular value at a chosen point. The overhead of this reduction is concretely very small for many practical computations [Mod24].

Polynomial evaluation step. More specifically, the GKR protocol reduces the verification of $C(\tilde{X}) = z$ to evaluating the expression

$$((1 - r'_1, r'_1) \otimes \cdots \otimes (1 - r'_{k'}, 1 - r'_{k'}))^T \tilde{X}((1 - r_1, r_1) \otimes \ldots (1 - r_k, r_k)), \tag{5}$$

where $r_1, \ldots, r_k \in \mathbf{F}$ and $r'_1, \ldots, r'_{k'} \in \mathbf{F}$ are some entries and, say, $\tilde{X} \in \mathbf{F}^{2^{k'} \times 2^k}$ has side lengths which are powers of two. (Indeed, these entries will correspond to the random indices of $\bar{g}_r$ and $\bar{g}_{r'}$; *cf.*, (1).) This final evaluation step is often a significant part of both the computational and communication cost of a succinct proof. Another important observation we use is that we may view

$$\tilde{X}((1 - r_1, r_1) \otimes \ldots (1 - r_k, r_k))$$

as a *partial evaluation* of the polynomial with coefficients given in $\tilde{X}$. More specifically, this is called a partial evaluation as, interpreting $\tilde{X}$ as the coefficients of some multilinear polynomial of $k + k'$ variables, this operation corresponds to partially evaluating the polynomial over the first k variables, $r_1, \ldots, r_k$, leaving the remaining variables (those corresponding to $r'_1, \ldots, r'_{k'}$) free.

Multilinear PCS. Another important set of protocols which we will use are protocols for multilinear evaluation based on interactive oracle proofs of proximity (IOPPs) such as WHIR [ACFY24], FRI-Binius [DP24a], and BaseFold [ZCF24]. (For the rest of the paper, we will use 'WHIR' as the standard protocol, but others would work equally well.) At a very high level, these protocols take a commitment to an encoded message $G\tilde{x}$, where $\tilde{x}$ is the message, and efficiently evaluate

$$\tilde{x}^T((1 - r_1, r_1) \otimes \ldots (1 - r_k, r_k)),$$

for some stated values $r \in \mathbf{F}^k$ by opening the encoding at a small number of places and performing some additional checks. In 'standard' WHIR, FRI-Binius, and BaseFold, a meaningful portion of the computational work for the prover is spent computing the initial encoding of the message $G\tilde{x}$. We will show that this initial computational work is already done by the encoder in its encoding step and can be very easily verified. Additionally, one important part of either of these schemes is that we may efficiently *batch* many evaluations. For example, given a number of messages stacked in the rows of a matrix $\tilde{T}$, we can efficiently prove that

$$\tilde{T}((1 - r_1, r_1) \otimes \ldots (1 - r_k, r_k)) = u_r.$$

Goal. What we will show next is that any sampler has essentially performed much of the work of evaluating expression (5) over the unique data matrix $\tilde{X}$ encoded by the encoder §2.4 and can, with only a small amount of overhead, also evaluate the multilinear polynomial whose coefficients correspond to a subset of the rows of $\tilde{X}$. The former is very useful for chains that control all contents of $\tilde{X}$ and wish to verify, say, a state transition function on input $\tilde{X}$ is executed 'correctly'. The latter is very useful for multiple rollups each of which post their data to specific rows of $\tilde{X}$ and wish to verify some computation over their specific subset of the data.

4 Complete evaluation

In the case where a complete evaluation is necessary (for example, in the case of 'monolithic chains' who control the complete contents of $\tilde{X}$ and have a known function C for verifying the state; *i.e.*, chains that wish to verify $C(\tilde{X}) = z$), a very simple amendment to the encoder algorithm and sampling procedure allows the sampler to evaluate (5) with no additional communication and computation. In other words, other than proving or verifying the GKR proof (minus the final polynomial evaluation), there is no additional communication or computation necessary to evaluate the polynomial (5). For simplicity, as in the presentation of §2, we will assume that $\tilde{X}$ has side lengths which are powers of two, $\tilde{X} \in \mathbf{F}^{2^k \times 2^{k'}}$.

4.1 Amendment to encoding and sampling

Here, we will show how to amend the encoding and sampling algorithms to allow for the efficient evaluation of (5).

Encoder. To do this, the encoder (or, equivalently, prover in this case) first commits to the supposed encoding of $\tilde{X}$ and runs the GKR protocol's proving step. (In other words, the encoder runs steps 1 and 2 of the encoder algorithm §2.2 and then runs the GKR protocol's proving step.) From the GKR protocol, using the Fiat–Shamir heuristic, the prover will receive some randomness which it then uses to run the rest of the protocol. From the matrix-vector product property (§2.4) and the fact that $\bar{g}_r$ are using logarithmic randomness, we

know that y_r is the partial evaluation of $\tilde{X}$:

$$y_r = \tilde{X}\bar{g}_r = \tilde{X}((1 - r_1, r_1) \otimes \ldots (1 - r_k, r_k)), \tag{6}$$

except with probability of error no more than that of (3).

Sampler. The sampler receives the GKR proof (without the final polynomial evaluation) and then verifies all steps of the sampling algorithm. Finally, in the last step of the sampling algorithm, note that the sampler has computed

$$\bar{g}_{r'}'^T y_r = ((1 - r_1', r_1') \otimes \cdots \otimes (1 - r_{k'}', 1 - r_{k'}'))^T y_{\bar{r}},$$

which, combined with (6), is exactly the complete evaluation (5).

Cost. Note first that the only additional cost over running the vanilla protocol of §2 is that of the intermediate steps of the GKR proof, which, for many practical circuits is concretely small in both communication and computation.

GKR circuits for base layers. In general, data availability blockchains such as Celestia attempt to remain as minimal as possible and support only the smallest subset of necessary computation needed to enable their use. For example, in the case of Celestia, most of the computation comes from verifying that payments have been made for included data, verification of IBC transfers [Cel25], and verification of signatures. These computations can, in principle, be expressed as data-parallel circuits, making them well-suited for GKR proofs with little extra overhead compared to directly verifying the computations. In the general setting, the data availability chain need not be aware of the specific computations being performed by the rollup. However, when the prover and the encoder are run by the same entity, data commitments can precede GKR proving, condensing the two constraints in step 5 above to one.

5 Evaluation over subset of rows

In this section, we will show how to 're-use' the work done in the encoding and sampling steps of the protocol to substantially lower the amount of work needed to construct a multilinear polynomial evaluation over a subset of the rows of $\tilde{X}$, as opposed to the whole matrix. Additionally, unlike in the previous amendment where we assumed the computation was known to the encoder when performing the encoding step (as is the case where, *e.g.*, a protocol is using its own data availability layer exclusively), in this amendment the encoder will not need to know anything about the statement (the function C) to be proven. This fact is very useful in the practical setting where, for example, many distinct protocols (or *rollups*) share the same 'large' data block $\tilde{X}$, whose availability is maintained by an external protocol. Each rollup does not need to read nor prove a statement over all of the data in the block $\tilde{X}$, only some subset of it, while the external protocol does not need to know

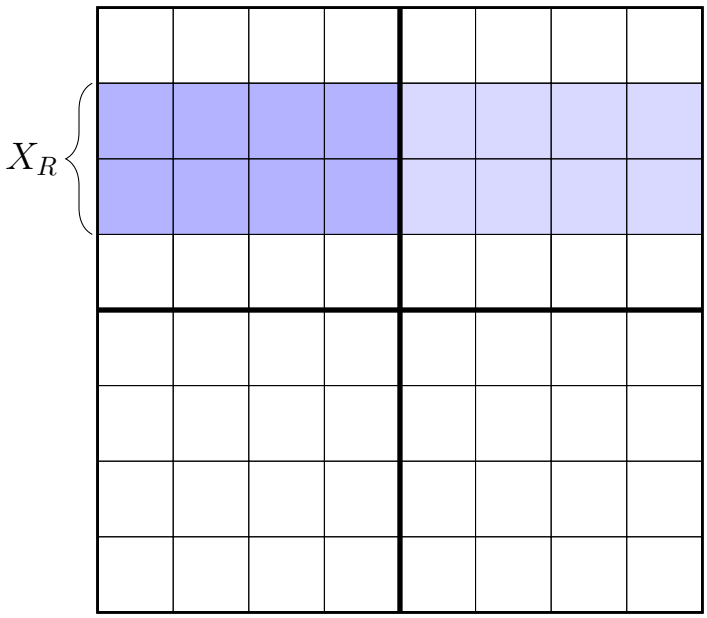

Figure 2: Subset of rows of interest. The lightly shaded region denotes an encoding of the rows of the data such that $X_R = \tilde{X}_R G'^T$.

anything about these operations other than ensuring the relevant data is correctly encoded and available.

5.1 Requirements

In this setting, a rollup has some predicate, usually some equality $C(\tilde{X}_R) = z$ over some subset of the rows $R \subseteq \{1, \ldots, n\}$, that must be satisfied. (For example, such a predicate might include constraints that require that balances posted to $\tilde{X}$ by the rollup are nonnegative and that transfers of assets between accounts on the rollup have been netted out correctly. See figure 2 for an illustration.) In this setting, the encoder is still present but there is an additional *rollup prover* (distinct from the encoder) who will seek to convince a *light client* that the predicate is satisfied and the data $\tilde{X}_R$ corresponds to the data which the encoder has encoded. As the name 'light client' suggests, the amount of data that can be downloaded by and computational power available on this client are limited and any algorithms it runs must be relatively lightweight.

Verifier requirements. The up prover must convince a light client that: (1) the rollup's relevant data $\tilde{X}_R$ can be uniquely recovered from the commitments provided by the encoder and (2) the predicate is indeed satisfied by this $\tilde{X}_R$ in that, say, $C(\tilde{X}_R) = z$. For now we will assume that the light client has run the sampling algorithm of §2.3 and therefore has the guarantees of $\tilde{X}$ provided in §2.4 with the bound of (3). (We will discuss later how to reduce these requirements for the light client.)

5.2 Protocol amendments

There are almost no amendments to the encoder and sampler algorithms, but we will define the behavior of the rollup prover and light clients in what follows.

Encoder and sampler. The encoder and sampler algorithms are identical to the ones described in §2.2 and §2.3, respectively.

Rollup prover. The rollup prover first verifies that the sampling algorithm passes (to ensure that the data is indeed correctly encoded) after receiving the logarithmic randomness $\bar{g}_r$, the vector y_r, and the commitment to Y as part of the protocol. It then runs the steps we present below. We will assume the data relevant to the rollup is in the $R \subseteq \{1, \ldots, n\}$ rows of $\tilde{X} \in \mathbf{F}^{n \times n'}$, the (unique) matrix known to exist from the sampling algorithm and, for simplicity, assume that the encoding is systematic such that $G_R \tilde{X} = \tilde{X}_R$.

1. Download the R rows of Y and verify that they are valid encodings; *i.e.*, $Y_R = \bar{Y}_R G'^T$
2. Verify that $\bar{Y}_R \bar{g}_r = G_R y_r$ (this will imply that $\bar{Y}_R = G_R \tilde{X}$, with high probability)
3. Run GKR to reduce verifying $C(\bar{Y}_R) = z$ to checking that $\bar{g}_{\bar{r}}^T \bar{Y}_R \bar{g}'_{\bar{r}'} = t$, for some $t \in \mathbf{F}$
4. Receive structured randomness $\bar{g}_{\bar{r}} \in \mathbf{F}^{|R|}$ and $\bar{g}'_{\bar{r}'} \in \mathbf{F}^{n'}$ from the final step of GKR
5. Run WHIR's proving step with two constraints (1) $\bar{Y}_R \bar{g}'_{\bar{r}'} = u_{r'}$ and (2) step 2's check (*i.e.*, two multilinear evaluation constraints, which can be efficiently batched)
6. Send the GKR and WHIR proofs, along with the output $u_{r'}$, to the light node

By the correct encoding guarantee given in §2.4 and the fact that G is systematic, we know that $\tilde{Y}_R = \tilde{X}_R$ with high probability. We note that the savings here come from the fact that one of the most expensive steps, namely the encoding of the rows of $\tilde{Y}_R = \tilde{X}_R$, has already been performed and can be easily verified by anyone who has already run the sampler. The rollup prover can then 'piggyback' off of the work already performed to reduce both the proof size and amount of computation needed to prove the statement $C(\tilde{Y}_R) = z$. In the case where Basefold [ZCF24] or FRI-Binius [DP24a] are used, the rollup prover's work as part of the PCS is linear time since the $n \log n$ component is outsourced to the DA encoder. Using WHIR [ACFY24], the rollup prover still has to compute encodings in subsequent folds, incurring $n \log n$ cost, but we expect the concrete difference to be small enough to justify a smaller proof size and more efficient verifier. Finally, we note that WHIR can capture arbitrary sumcheck-like constraints, but we do not exploit this flexibility here. For example, it may be possible, using this observation, to further reduce overhead by incorporating GKR's final sumcheck layer into the WHIR constraint, evaluated directly on the input layer.

Light node. We assume that the light node has also first verified that the sampling algorithm has passed. (It can do this by running the usual sampling algorithm, or, *e.g.*, by receiving a recursive proof that the sampling algorithm was correctly performed.) The light node then simply verifies that the received proofs given by the rollup prover are correct for the provided inputs (which it has available as it has verified that the sampling algorithm has passed) and then simply verifies that $\bar{g}_{\bar{r}}^T u_{\bar{r}'} = t$ to verify that the claimed evaluations are equal.

5.3 Discussion

Inclusion and hashing. Using this simple construction, we note that the light node is convinced of both inclusion of the input data (enforced by the proof generated in step 6 of the rollup prover's algorithm) and correctness of execution. Since a meaningful portion of the WHIR proving time, especially for large folding factors, is spent constructing and committing to the first oracle, the prover reduces proving time significantly (by outsourcing the top layer to the DA encoder). Additionally, the prover does not need to compute inclusion proofs against the encoded data square by computing hashes in-circuit as in existing protocols [Sov24] as inclusion is guaranteed by the checks done by the sampling algorithm. Since protocols such as Celestia [Cel24] use hash functions that are expensive to prove, this provides additional non-negligible savings for the rollup prover and the light node.

Zero-change. Another important observation involves that this protocol is essentially zero-change for any data availability layer that already performs tensor encodings. In particular, any party can compute the corresponding ZODA proofs given by y_r and $w_{r'}$, including the rollup prover, for, *e.g.*, data availability layers that do not have succinctly verifiable encodings. The rollup prover is also, of course, independent of the encoder and samplers, who can continue running the original protocol and the additional proofs (which are concretely small) can be sent out-of-band for clients who wish to accept them.

6 Conclusion

We have shown how to reuse encoding and sampling work as part of a data availability construction to substantially reduce the work required to prove the validity of a blockchain's state transition function. When the state transition function is defined over all data posted to the data availability layer, we have shown how to entirely remove work associated with committing and opening polynomials by leveraging the work done during the encoding process of the data availability layer. When the state transition function depends on a subset of the data in the data availability layer, we have shown how to substantially reduce the work necessary to commit and open the polynomial associated with this subset of the data. We expect that this latter construction will be useful for many rollups and other blockchains that share data availability layers.

Future work. While our construction utilizes the GKR protocol, it is plausible that the insights of this work may apply to other constructions and arithmetizations. The insights are, for example, directly applicable to other succinct proof systems that require commitments only to the inputs to the computation, such as the IP for matrix multiplication [Tha22, Sec. 4.4]. Similar techniques may apply to other constructions and arithmetizations as well, though we do not explore these here.

Acknowledgments

The authors would like to thank Preston Evans for carefully reading an early draft of this paper and suggesting a number of changes, many of which we incorporated. The authors would also like to thank Ryan Cao and Vishruti Ganesh for carefully and patiently explaining the mechanics of GKR, along with Steven Smith, Giorgos Zirdelis, and Daniel Shorr for providing feedback on an early presentation of the results of this paper. The authors would like to thank Nicolas Mohnblatt and Giacomo Fenzi for helpful conversations on WHIR and its construction along with Benjamin E. Diamond for a careful reading of the preprint.

References

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed–solomon proximity testing with super-fast verification. Cryptology ePrint Archive, Paper 2024/1586, 2024.

[AER24] Guillermo Angeris, Alex Evans, and Gyumin Roh. A note on Ligero and logarithmic randomness. Cryptology ePrint Archive, Paper 2024/1399, 2024.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2087–2104, Dallas Texas USA, October 2017. ACM.

[ASB18] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities. *CoRR*, abs/1809.09044, 2018.

[Cel24] Celestia Documentation. Celestia's data availability layer, 2024. Accessed: 14 September 2024.

[Cel25] Celestia Docs. *IBC relaying guide*, 2025. Retrieved January 20, 2025.

[DG24] Benjamin E. Diamond and Angus Gruen. Proximity gaps in interleaved codes. Cryptology ePrint Archive, Paper 2024/1351, 2024.

[DP24a] Benjamin E. Diamond and Jim Posen. Polylogarithmic proofs for multilinears over binary towers. Cryptology ePrint Archive, Paper 2024/504, 2024.

[DP24b] Benjamin E. Diamond and Jim Posen. Proximity testing with logarithmic randomness. *IACR Commun. Cryptol.*, 1(1):2, 2024.

[EA23] Alex Evans and Guillermo Angeris. Succinct proofs and linear algebra. Cryptology ePrint Archive, Paper 2023/1478, 2023.

[EMA24] Alex Evans, Nicolas Mohnblatt, and Guillermo Angeris. ZODA: Zero-overhead data availability. Cryptology ePrint Archive, Paper 2025/034, 2024.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 113–122, New York, NY, USA, 2008. ACM.

[HSW23] Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Foundations of data availability sampling. Cryptology ePrint Archive, Paper 2023/1079, 2023.

[HSW24] Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. FRIDA: data availability sampling from FRI. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 289–324. Springer, 2024.

[Mod24] Modulus Labs. Scaling intelligence: Verifiable decision forest inference with remainder, 03 2024. Unpublished manuscript.

[Sov24] Sovereign Labs. Sovereign SDK: Data availability specification, 2024. Accessed: 29 December 2024.

[Tha22] Justin Thaler. Proofs, Arguments, and Zero-Knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. Cryptology ePrint Archive, Paper 2019/317, 2019.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. Basefold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part X*, page 138–169, Berlin, Heidelberg, 2024. Springer-Verlag.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. Cryptology ePrint Archive, Paper 2017/1145, 2017.

A Proof of ZODA variation

From before, we assume that readers are familiar with the log-randomness variation [DP24b, AER24] of the Ligero subspace distance check [AHIV17]. We will also assume some familiarity with the ZODA [EMA24] protocol. Though not strictly necessary, the proofs here are very similar in spirit to the proofs given in [EMA24, App. E] with only small changes, but we put them here in full for completeness.

A.1 (Short) preliminaries

We fix some basic notation and implications thereof in what follows. For more, see, *e.g.*, [EA23, AER24] which we reference in what follows.

Probabilistic implications. We use the probabilistic implication framework of [EA23] to significantly simplify the notation. In particular, we use the following definition and its consequence. Letting P_r and $Q_{r'}$ be statements which are either true or false (*i.e.*, Boolean variables), then we write

$$P_r \quad \underset{p}{\Longrightarrow} \quad Q_{r'},$$

whenever $\Pr(P_r \wedge \neg Q_{r'}) \leq p$, over the randomness r, r' drawn from a provided distribution, which will always be stated in the text. We call p the *error probability*, which means that the statement above may be read as: if P_r is true, then $Q_{r'}$ must also be true, except with (error) probability no larger than p. ('Standard' logical implications are exactly those for which the probability of error is 0.) This notation serves to clean up proofs since it is very similar to the 'standard' logical rules. For example, much like logical implications, probabilistic implications may also be chained, with some additional error.

$$P_r \quad \underset{p}{\Longrightarrow} \quad Q_{r'}, \qquad \text{and} \qquad Q_{r'} \quad \underset{p'}{\Longrightarrow} \quad T_{r''},$$

over randomness r, r' and r', r'', then

$$P_r \quad \underset{p+p'}{\Longrightarrow} \quad T_{r''}.$$

(This follows essentially from a union bound argument.) Similarly, if we have the deterministic statements $Q^{(1)}$, $Q^{(2)}$ and the following probabilistic implications are true

$$P_r^{(1)} \quad \underset{p}{\Longrightarrow} \quad Q^{(1)} \qquad \text{and} \qquad P_r^{(2)} \quad \underset{p}{\Longrightarrow} \quad Q^{(2)}, \tag{7}$$

then

$$P_r^{(1)} \wedge P_r^{(2)} \quad \underset{p}{\Longrightarrow} \quad Q^{(1)} \wedge Q^{(2)}.$$

We use both observations in the statements and proofs below.

Subspace distance check. The main checks performed in the sampling protocol depend on the subspace distance check for logarithmic randomness, originally derived in [DP24b] as an extension of the Ligero subspace distance check [AHIV17], though we use the notation and improvement of [AER24]. As in §2, define

$$\bar{g}_r = (1 - r_1, r_1) \otimes \cdots \otimes (1 - r_k, r_k),$$

where $r_1, \ldots, r_k \in \mathbf{F}$ uniformly drawn from $\mathbf{F}$ and $\bar{g}_r \in \mathbf{F}^{2^k}$. As before, we have some code $G \in \mathbf{F}^{m \times n}$ with distance $d > 0$. The *subspace distance check* states that, for a given matrix $X \in \mathbf{F}^{m \times 2^k}$ and some distance $q < d/3$ the following probabilistic implication holds:

$$\|X\bar{g}_r - Gy_r\| < q \quad \underset{p}{\Longrightarrow} \quad \text{there exists } \tilde{X} \in \mathbf{F}^{n \times 2^k} \text{ such that } \|X\bar{g}_r - G\tilde{X}\| < q, \tag{8}$$

where y_r can depend on the randomness r and the probability of error p satisfies

$$p \leq \frac{kq}{|\mathbf{F}|}, \tag{9}$$

over the uniform randomness $r \in \mathbf{F}^k$. (It is very important to note that the matrix $\tilde{X}$ is independent of the randomness r. We will also assume that q is an integer for simplicity but, if not, simply replace q with $\lceil q \rceil$ in the error bound.) In other words, if we take a linear combination of the columns of X via the structured random vector $\bar{g}_r$ and the result is q-close to some codeword Gy_r, then it must be true that the original matrix X must be q-close to some (fixed) correctly-encoded matrix $\tilde{X}$, except with error probability no more than p. We will use this check many times in what follows.

A.2 Sampler properties

We will show that, if the checks of the sampler algorithm (§2.3) succeed, then it must have been the case that the (opaque) matrix Y, corresponding to the rows of Z when the encoder is honest, must be close to the tensor encoding of some matrix $\tilde{X}$. Similarly it must have been true that the (opaque) matrix W, corresponding to the columns of Z when the encoder is honest, must be close to the transpose of the tensor encoding of the *same matrix* $\tilde{X}$. Along the way, we will show the matrix-vector product property and the correct encoding property hold, since these will be useful in establishing the unique decoding property. With that in mind, along with the subspace distance check, the proof is as follows.

Matrix-vector product. First, note that step 6 simply checks that two m-vectors match at $|S|$ uniformly randomly picked indices. In other words, if the vectors differ in at least q entries, the probability step 6 succeeds is no more than

$$\left(1 - \frac{q}{m}\right)^{|S|}.$$

Writing this out in probabilistic implications, this is simply saying that, for every $r \in \mathbf{F}^k$, the following is true:

$$\bar{Y}_S \bar{g}_r = G_S y_r \quad \underset{p'}{\Longrightarrow} \quad \|\bar{Y}_S - G y_r\| < q, \tag{10}$$

where the randomness is over the chosen indices $S \subseteq \{1, \ldots, m\}$ and the error satisfies the following bound:

$$p' \leq \left(1 - \frac{q}{m}\right)^{|S|}. \tag{11}$$

The conclusion of (10) is the proposition of the subspace distance check (8), which means that, by chaining probabilistic implications, we know

$$\tilde{Y}_S \bar{g}_r = G_S y_r \quad \underset{p+p'}{\Longrightarrow} \quad \text{there exists some matrix } \tilde{Y} \text{ such that } \|\bar{Y} - G\tilde{Y}\| < q. \tag{12}$$

From before, the randomness is over uniformly random $S \subseteq \{1, \ldots, m\}$ and independent, uniformly sampled $r \in \mathbf{F}^k$. By the triangle inequality and the definition of the 'norm' $\|\cdot\|$,

$$\|G(\bar{Y}\bar{g}_r - y_r)\| \leq \|(Y - G\bar{Y})\bar{g}_r\| + \|Y\bar{g}_r - G y_r\| \leq \|Y - G\bar{Y}\| + \|Y\bar{g}_r - G y_r\| < 2q \leq d.$$

Since the distance of G is (at least) d, then we have that

$$\tilde{Y}\bar{g}_r = y_r, \tag{13}$$

with error probability no more than $p + p'$, from (12). (This is essentially the matrix-vector product property, except we still need to show that this $\tilde{Y}$ is consistent with the matrix derived from W. We do so later this section.)

Correct encoding property. This, in turn, implies that

$$G_S \tilde{Y} \bar{g}_r = G_S y_r = \bar{Y}_S \bar{g}_r,$$

where the second equality is the antecedent of (10). Finally, it is not hard to show that for any given vector $x \in \mathbf{F}^{2^k}$, we have that

$$\bar{g}_r^T x = 0 \quad \underset{p''}{\Longrightarrow} \quad x = 0, \tag{14}$$

where $p'' \leq k/|\mathbf{F}|$. (This can be slightly improved, see [EA23, §3.1.3], but we do not do so here.) Applying this probabilistic implication and using the conjunction property (7), we have that

$$G_S \tilde{Y} \bar{g}_r = \bar{Y}_S \bar{g}_r \quad \underset{p''}{\Longrightarrow} \quad G_S \tilde{Y} = \bar{Y}_S,$$

over the randomness $r \in \mathbf{F}^k$, sampled uniformly randomly.

Unique encoding for a single matrix. Finally, note that, by the check of step 5, we know

$$Y_S = \bar{Y}_S G'^T = G_S \tilde{Y} G'^T.$$

This, in turn, implies

$$\|Y - G\tilde{Y}G'^T\| < q,$$

as required. (Note that by the conjunction (7), we incur no additional error.) In English, we have shown that, by checking steps 5 and 7, the sampler may conclude that there exists some matrix $\tilde{Y}$ such that

$$\|Y - G\tilde{Y}G'^T\| < q,$$

except with error probability no larger than $p + p' + p''$. A nearly identical proof shows the same is true over W^T after checks 5 and 7, which gives

$$\|W^T - G'\tilde{W}G^T\| < q,$$

with the same error probability $p + p' + p''$ (assuming $|S'| = |S|$, the extension here being obvious). From before, the conjunction property (7) means that we do not incur additional error in concluding that both

$$\|Y - G\tilde{Y}G'^T\| < q \quad \text{and} \quad \|W^T - G'\tilde{W}G^T\| < q,$$

i.e., the error is no larger than $p + p' + p''$.

Consistency of unique encodings. Finally, the check in step 9 means that

$$\bar{g}'^T_{r'} y_r = w^T_{r'} \bar{g}_r.$$

Combining this with (13), we have that

$$\bar{g}'^T_{r'}(\tilde{Y}\bar{g}_r) = (\tilde{W}\bar{g}'_{r'})^T \bar{g}_r.$$

Or, after rewriting:

$$\bar{g}'^T_{r'}(\tilde{Y} - \tilde{W}^T)\bar{g}_r = 0.$$

Finally, applying the check (14) twice, we get

$$\bar{g}'^T_{r'}(\tilde{Y} - \tilde{W}^T)\bar{g}_r = 0 \quad \underset{2p''}{\Longrightarrow} \quad \tilde{Y} = \tilde{W}^T.$$

Chaining all implications together implies that, after the sampling algorithm is run, the sampler knows, by setting $\tilde{X} = \tilde{Y} = \tilde{W}^T$, that there exists some matrix $\tilde{X}$ such that

$$\|Y - G\tilde{X}G'^T\| < q \quad \text{and} \quad \|W^T - G'\tilde{X}^T G^T\| < q,$$

except with probability no more than $p + p' + 3p''$, over the randomness $r, r' \in \mathbf{F}^{2^k}$. Plugging in the inequalities for p in (9), p' in (11), and p'' defined after (14), we get

$$p + p' + 3p'' \leq \frac{k(q+3)}{|\mathbf{F}|} + \left(1 - \frac{q}{m}\right)^{|S|},$$

assuming $|S| = |S'|$ and $q = q'$, which is what is given in (3) after setting $q = d/3$.