ZODA: Zero-Overhead Data Availability

Alex Evans
aevans@baincapital.com

Nicolas Mohnblatt
nmohnblatt@baincapital.com

Guillermo Angeris
gangeris@baincapital.com

December 2024

Abstract

We introduce ZODA, short for 'zero-overhead data availability,' which is a protocol for proving that symbols received from an encoding (for tensor codes) were correctly constructed. ZODA has optimal overhead for both the encoder and the samplers. Concretely, the ZODA scheme incurs essentially no incremental costs (in either computation or size) beyond those of the tensor encoding itself. In particular, we show that a slight modification to the encoding scheme for tensor codes allows sampled rows and columns of this modified encoding to become proofs of their own correctness. When used as part of a data availability sampling protocol, both encoders (who encode some data using a tensor code with some slight modifications) and samplers (who sample symbols from the purported encoding and verify that these samples are correctly encoded) incur no incremental communication costs and only small additional computational costs over having done the original tensor encoding. ZODA additionally requires no trusted setup and is plausibly post-quantum secure.

Introduction

A proof of encoding is a protocol for convincing a sampler (who samples small chunks of a certain piece of data) that an encoder has correctly encoded some data. These protocols are used in a applications including verifiable information dispersal [Rab89, CT05, NNT22, FLLY24] and data availability sampling [ASB18, HSW23]. Such applications require highly efficient encoders and samplers, as well as minimal trust assumptions.

Data availability sampling. In this paper, we are primarily motivated by the application of data availability sampling (DAS), a notion first introduced in [ASB18]. A DAS protocol aims to convince samplers that a piece of data, held by nodes in a potentially lossy or adversarial network, can be recovered. In order to avoid downloading the data in its entirety, nodes instead request (an ideally small number of) random samples and make additional assumptions about the network. For example, nodes may assume that a certain number of honest nodes that are connected to the network are also requesting random samples, such that, if all sampling nodes come together, they can decode the original piece of data. These DAS protocols are primarily deployed to scale blockchains: nodes only need to download small subsets of the data, while maintaining strong availability guarantees. As a result, blockchains employing DAS techniques can theoretically support significantly larger blocks, since very few parties (if any) need to hold all of the data at once. Efficient DAS constructions rely on the properties of erasure codes. First, data is encoded using a suitable erasure code. Nodes then randomly sample symbols of the resulting codeword and accept the data as available if the network successfully responds to their queries. In order for nodes to be able to reconstruct data from enough samples, the received samples must be constructed correctly and nodes must collectively sample enough correctly-encoded symbols to reconstruct the original data. In this work, we are motivated by efficient protocols which show that the sampled encodings were correct, ideally with minimal overhead over sending just the sample. (For a broader treatment of DAS protocols, the works of [ASB18, HSW23] introduce several foundational definitions and primitives.)

Existing constructions. A number of protocols for proving that an encoding was constructed correctly have been proposed. One construction involves fraud-proofs [ASB18]. This construction incurs no encoder overhead but requires some nodes to redundantly recompute the encoding such that a fraud proof can be constructed if the encoder behaved dishonestly. This, in turn, reduces the efficiency of the resulting protocol and introduces a trust assumption—every node must be connected to at least one honest node who recomputes this complete encoding. Another category of protocols instead use validity proofs. The most popular such construction [But20] is based on the polynomial commitment scheme of [KZG10]. This construction is efficient to verify and removes the need for some nodes to always recompute the complete encoding but requires a trusted setup, is not post-quantum secure, and imposes large overhead on the encoder. This overhead comes from the fact that the encoder has to construct a KZG commitment to the encoding and open this commitment at every point which could be sampled in the associated DAS protocol. These computations are concretely expensive, even on modern hardware. A recently-proposed set of alternatives [HSW23, HSW24] adapt hash-based interactive oracle proofs which are plausibly post-quantum secure and do not require a trusted setup. Compared to the KZG scheme, these constructions feature efficient encoders but impose prohibitive costs on samplers: every sampler needs to download an (identical) large commitment. This commitment represents pure overhead for the network as it cannot be used to decode the original data and is only used to verify that the samples are correctly encoded.

This work. In this paper, we propose ZODA, a simple protocol for proving that received samples from a piece of data were correctly encoded. ZODA is extremely efficient for both the encoder and sampler and does not require a trusted setup. Specifically, the encoder incurs essentially no additional overhead beyond the cost of encoding the data and committing to the encoding (for example, using a Merkle tree). Concretely, the computational cost is nearly identical to running an encoder in the fraud-proof based construction of [ASB18]. Sampling communication is optimal in the sense that no additional data or proofs are required to establish the validity of the encoding beyond downloading parts of the encoding itself. Finally, we incur no network overhead beyond the transmission of the encoded data (and associated Merkle openings).

Comparison. Our construction amends the encoding procedure of [ASB18] to derive randomness from a partially-encoded matrix before completing the tensor encoding. As in the Ligero-based protocol of [HSW23], ZODA samplers uniformly sample rows and columns and perform consistency checks. However, in our resulting DAS construction, sampled rows and columns also double as valid samples for decoding, eliminating overhead for the sampler. This slight amendment to the encoding procedure of [ASB18] introduces negligible overhead for the encoder while removing the need for fraud proofs to ensure the encoding was constructed correctly. While we do not explicitly construct a DAS scheme which uses ZODA, we anticipate the scalability benefits to be significant. By removing the need for redundant re-encoding, ZODA can be used to reduce latency and network bandwidth compared to fraud-proof-based constructions. Compared to KZG-based constructions, we anticipate ZODA could meaningfully reduce proving latency and costs associated with constructing opening proofs, allowing the network to support more data while lowering both latency and hardware requirements.

Organization. In §1, we introduce some basic notation, which will considerably simplify proofs later in the paper, and some basic results we use throughout. In §2, we discuss subspace distance checks, a key component of the ZODA protocol, which we then present in §3, along with proofs that this protocol satisfies some basic guarantees. In §4, we give a concrete instance of the protocol, along with associated computational and communication costs for both the encoding and sampling algorithms in a variety of parameter settings. Finally, in §5, we present some practical extensions and optimizations that may be useful depending on the requirements of the application.

1 Notation, conventions, and preliminaries

Throughout this paper, we use the conventions and notation of [EA23], many of which are standard linear algebraic conventions. Let $\mathbf{F}$ be a (in this paper, finite) field and write $x \in \mathbf{F}^n$ to be a (column) n-vector with entries in $\mathbf{F}$. We write $A \in \mathbf{F}^{m \times n}$ for an $m \times n$ matrix A with elements in the field $\mathbf{F}$ and write $\mathcal{R}(A)$ as the set of all linear combinations of the columns of A, called its *range*. When convenient, we will also interpret the n-vector x to be an $n \times 1$ matrix and we use this in what follows. Finally, if $S \subseteq \{1, \dots, m\}$ then we write A_S as the (sub)matrix containing rows S, taken from A, such that

$$(Ax)_S = A_S x. \tag{1}$$

(We assume some standard ordering such that the left-hand-side and the right-hand-side make sense and are equal for any choice of A, S, and x.)

Weight. We define the *weight* of a matrix $A \in \mathbf{F}^{m \times n}$ to be

$$\|A\| = \text{number of nonzero rows of } A.$$

The weight $\|x\|$ of a vector $x \in \mathbf{F}^n$, interpreted as an $n \times 1$ matrix, is then the number of nonzero entries of x. A very convenient fact we will use later in this paper is that

$$\|Ax\| \leq \|A\|, \tag{2}$$

for any matrix $A \in \mathbf{F}^{m \times n}$ and vector $x \in \mathbf{F}^n$.

Codes and distance. We define a linear *code* to be a matrix $G \in \mathbf{F}^{m \times n}$. (Technically, codes are usually defined to be linear subspaces, but we ignore the distinction here as it is not useful in our case.) The *distance* $d \geq 0$ of the code G is defined as

$$d = \min_{x \neq 0} \|Gx\|.$$

In other words, the distance is defined as the smallest weight of any possible codeword Gx generated by nonzero messages x. The distance of a vector space V is defined as the distance of any injective matrix G' which generates it, *i.e.*, $V = \mathcal{R}(G')$. An important implication of this definition of distance, which we use in what follows, is that a matrix G is injective if, and only if, its distance d is strictly positive, *i.e.*, if $d > 0$. Though it will only be mentioned in passing, we define the *relative distance* of G to be d/m, the ratio of the distance and the length of the codewords from G, and its *rate*, usually written ρ, is defined as the message size over the codeword length, $\rho = n/m$.

Probabilistic implications. Finally, we find it convenient to use the *probabilistic implications* of [EA23]. Let P_r and $Q_{r'}$ be logical statements depending on some randomness r and r' drawn from some suitable distribution, which will always be specified in the text. (Though most of the time we will have $r = r'$ or r and r' independent.) We write,

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

whenever $\Pr(P_r \wedge \neg Q_{r'}) \leq p$. In other words, the above statement says that the probability that the conclusion is false, provided the premise is true, is no more than p over the provided randomness. One useful consequence of this notation, which will be used consistently throughout, is the following. Given statements P_r, $Q_{r'}$, and $T_{r''}$, if

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

over randomness r and r' in the first, and in the second over randomness r' and r'', then we have the following probabilistic implication:

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

over the randomness r and r'' (*i.e.*, marginalizing out r' in the resulting distribution). This follows nearly directly from the union bound. Note that this is true independent of the distribution from which r, r', and r'' are drawn from. For a proof, see [EA23, App. A].

Sparsity checks. The notion of a *sparsity check* will also be used throughout this paper. Provided a vector $x \in \mathbf{F}^n$ and some sparsity parameter $q > 0$, a sparsity check is a probabilistic implication of the form

$$x_S = 0 \quad \underset{p}{\Longrightarrow} \quad \|x\| < q, \tag{3}$$

where $S \subseteq \{1, \dots, n\}$ is a uniformly random subset of size $|S|$, and

$$p \leq \left(1 - \frac{q}{n}\right)^{|S|}. \tag{4}$$

(If $|S|$ is on the order of n, a better bound is possible, though this bound will suffice for our purposes.) Sparsity checks are particularly useful for verifying that two vectors are 'close' in that they differ in at most q entries. They are also useful since, given a matrix A, we can check that Ax is close to y by checking that

$$(Ax)_S = y_S,$$

which only requires access to the S rows of A, using (1), along with the S entries of y. We will constantly be using the probability p provided in (3) and its bound (4) throughout the rest of the paper and proofs.

2 Subspace distance checks

We assume reasonable familiarity with the Ligero protocol [AHIV17] and associated proximity tests throughout the remainder of the paper. For a gentle introduction, using the notation and conventions of this paper, we suggest [AER24]. We begin with a basic overview of subspace distance checks, which we use extensively in the construction of the protocol.

2.1 Subspace distance checks

In what follows, $G \in \mathbf{F}^{m \times n}$ will denote a code with distance $d > 0$ with $V = \mathcal{R}(G)$, and $\bar{G} \in \mathbf{F}^{\bar{m} \times n}$ denotes another code, which we define next. We write $\bar{g}_{\bar{r}}^T$ for the $\bar{r}$th row of $\bar{G}$.

Subspace distance check. At the highest level, a *subspace distance check*, sometimes also known as a *proximity test*, seeks to evaluate whether a given matrix $X \in \mathbf{F}^{m \times n'}$ 'is near' to an encoding of some matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$. In particular, a proximity test makes the following guarantee:

$$\|X\bar{g}_{\bar{r}} - V\| < q \quad \underset{p'}{\Longrightarrow} \quad \|X - G\tilde{X}\| < q,$$

where $\bar{r}$ is uniformly drawn from $1, \dots, \bar{m}$, the proximity parameter $q \leq d/2$ is user-set, and the error probability p' depends on the exact choice of code $\bar{G}$ used. (We will make this concrete in what follows.) In other words, a subspace distance check says, roughly: if a (structured) random linear combination of the columns of X is q-close to some vector space V, generated by the code G, then the columns of the matrix of X must be close to the encoded columns of some matrix $\tilde{X}$. This reduces checking that each column of a matrix X is close to a vector space V, to checking that a single vector is close to the vector space V.

Known results. This beautiful fact was first discovered in [AHIV17] where $\bar{G}$ is the Hadamard code when $q \leq d/3$ and G is otherwise arbitrary, with $p' \leq q/|\mathbf{F}|$. It was then extended to the case where G is a Reed–Solomon code and $\bar{G}$ is either Hadamard or Reed–Solomon by [BCI+23] in the case where $q \leq d/2$; here the bounds are $p' \leq m/|\mathbf{F}|$, if Hadamard, or $p' \leq mn/|\mathbf{F}|$, if Reed–Solomon. It has been further extended to the case where $\bar{G}$ has structured 'logarithmic randomness' in [DP24] (*i.e.*, when $\bar{G}$ can be written as the Kronecker product of generators of Hadamard codes for messages of length 2) and G is any matrix with distance d, for $q < d/3$ and $p' \leq 2q/|\mathbf{F}|$. This bound was improved and generalized for any $\bar{G}$ that can be written as the Kronecker product of any k matrices $\bar{G}_i$, each with 2 columns, for $q \leq d/3$ and error probability $p' \leq q((1-\delta_1) + \cdots + (1-\delta_k))$ where δ_i is the relative distance of matrix $\bar{G}_i$ in [AER24]. And, finally, it was shown that for certain structured codes G, such as Reed–Solomon, this last bound can be strengthened from requiring that $q \leq d/3$ to any case where $q \leq d/2$, in [DG24]. While there are many nice results for these codes, we will only use two in the rest of the paper, which we describe next in full detail.

2.2 Subspace distance checks in this paper

In this paper, we will use two important special cases of the subspace distance checks. Here, the matrices $G \in \mathbf{F}^{m \times n}$ (with distance d) and $\bar{G}$ (defined soon) along with the space $V = \mathcal{R}(G)$ are defined similarly as above, along with the matrix $X \in \mathbf{F}^{m \times n'}$ and $\tilde{X} \in \mathbf{F}^{n \times n'}$.

Subspace distance for general codes. The main check we will use is the original construction of Ligero [AHIV17]. Here, we define $\bar{G} \in \mathbf{F}^{|\mathbf{F}|^n \times n}$ to be the Hadamard code; *i.e.*, the matrix whose rows contain all possible n-vectors with elements in $\mathbf{F}$. Drawing a uniformly random row $\bar{g}_{\bar{r}}$ from $\bar{G}$ corresponds to drawing a uniformly random n-vector with elements in $\mathbf{F}$. In this case, for any vector space V with distance d we have, for any $q \leq d/3$,

$$\|X\bar{g}_{\bar{r}} - V\| < q \quad \underset{p'}{\Longrightarrow} \quad \|X - G\tilde{X}\| < q, \tag{5}$$

where $\bar{r} \in \{1, \dots, |\mathbf{F}|^n\}$ uniformly drawn and

$$p' \leq \frac{q}{|\mathbf{F}|}.$$

In English, the probabilistic implication (5) roughly says that: if a uniformly random linear combination of the columns of X is q-close to some vector space V, then the columns of X must each have been q-close to the vector space, except with error no more than p'. Since $q \leq d/3$ is often set as large as possible, then the error bound becomes

$$p' \leq \frac{d}{3|\mathbf{F}|}. \tag{6}$$

Subspace distance for Reed–Solomon codes. In the special case that $G \in \mathbf{F}^{m\times n}$ is a Reed–Solomon code matrix and V is its corresponding range, we have a slightly more relaxed regime. From before, let $\bar{G}$ be the Hadamard code for $\mathbf{F}$, then we can set $q \leq d/2$ such that

$$\|X\bar{g}_{\bar{r}} - V\| < q \quad \underset{p'_{\text{rs}}}{\Longrightarrow} \quad \|X - G\tilde{X}\| < q,$$

where $\bar{r}$ is uniformly drawn from $\{1, \dots, |\mathbf{F}|^n\}$, and

$$p'_{\text{rs}} \leq \frac{m}{|\mathbf{F}|}. \tag{7}$$

For comparison's sake, let $m = 2n$. Since we often set $q \leq d/2$ and Reed–Solomon codes are maximum distance separable, *i.e.*, satisfy $n = m - d + 1$, then the error bound in this case is only slightly worse than the original bound (6):

$$p'_{\text{rs}} \leq \frac{2(d-1)}{|\mathbf{F}|}$$

by a factor of about 6. The trade-off is generally worth it since fewer queries are required to ensure that $\|X\bar{g}_{\bar{r}} - V\| < d/2$ versus $d/3$ to any fixed error probability, while only gaining a very small constant factor of error.

Note. We will be using p' as defined in (5) constantly throughout the paper (along with p, as defined in (3) and after) in both proofs and discussions. Since these will be so commonly used, we will reserve the symbols p and p' to be exactly as defined here, unless otherwise specified in the text.

3 Zero-overhead proofs of encoding for tensor codes

In this section, we present a description of the protocol in the interactive setting. We will show that a simple modification to an encoding procedure for the tensor code $G' \otimes G$ where

$G \in \mathbf{F}^{m \times n}$ and $G' \in \mathbf{F}^{m' \times n'}$ results in a codeword whose rows and columns are succinct proofs of their own correct encoding. In particular, we will show that sampling a small number of rows and columns of this purported codeword convinces a sampler that (a) the codeword is, with high probability, very close to a correct encoding of some unique message, and (b) the received rows and columns are correct encodings of this unique message. (We call such a scheme a *proof of encoding* since it shows that the received rows and columns are correct encodings of some unique message.) It will also have one important additional property: any sampler convinced of (a) and (b) who receives further rows and columns of the codeword can check to see if these rows and columns are correct encodings, using no additional communication and very little additional computation, even if these rows and columns are selected adversarially; *i.e.*, after observing the sampler's randomness. We call this construction *zero-overhead* since the resulting encoding requires no more space (and only marginally more computation) over having done the standard tensor encoding.

High-level idea. A tensor encoding of some data matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$ is given by

$$G\tilde{X}G'^T.$$

The idea for a zero-overhead proof of encoding is to add a small amount of randomness to the encoding, given by some diagonal matrix $D_{\bar{r}}$ with uniformly random entries in the diagonal:

$$G\tilde{X}D_{\bar{r}}G'^T.$$

(Here, the randomness may be interactive or derived from a certain commitment, as we describe next.) Surprisingly, this small change allows anyone who samples a small number of rows and columns of this encoded matrix to verify that, with high probability, these rows and columns are indeed correct encodings of some $\tilde{X}$, without having to download the complete matrix. In other words, this modification turns a tensor code's rows and columns, roughly, into their own succinct proofs which prove that they must have been correctly encoded. (There are some slight complications with this exact picture, as described here, as G and G' must be systematic in this construction, but the high-level idea is the same.)

Notation. For notational convenience, let g_i^T denote the ith row of G and $\tilde{g}'_r$ denote the rth *column* of G'. As in the previous section, §2.2, we will define $\bar{G} \in \mathbf{F}^{\bar{m} \times n'}$ to be the Hadamard code matrix such that $\bar{m} = |\mathbf{F}|^{n'}$ and $\bar{g}_{\bar{r}}^T$ will denote its $\bar{r}$th row. This notation, and its associated dimensions, will remain consistent for the rest of the paper unless otherwise stated.

3.1 Protocol description

The protocol, which we call ZODA, short for zero-overhead data availability, is broken down into three algorithms, each providing certain guarantees. The first is the encoding algorithm. This algorithm takes some data, represented as a matrix, and constructs an encoding of it such that rows and columns of this encoding are themselves proofs of their own correct encoding, assuming the protocol was followed. The second is the sampling algorithm. The sampling algorithm allows anyone who is sampling a small number of rows and columns of a purported encoding to verify 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. Finally, the decoding algorithm allows anyone who has (1) run the sampling protocol, and (2) subsequently received 'enough' rows and columns of the purported encoding, in a way we make precise next, to check that these received rows/columns were correctly encoded and, if enough of these checks pass, to recover the unique encoded message by using any erasure decoder for G or G'. Though this protocol is written with an eye towards its use in data availability, we do not talk about specific applications except at a high level in this paper. For (much) more, including definitions for the specific case of data availability, and other constructions used there, see, [ASB18] and [HSW23].

3.1.1 Encoding algorithm

In this algorithm, the encoder will (a) commit to an encoding $G\tilde{X}$ and then (b) receive a uniformly random $\bar{r} \in \{1, \dots, \bar{m}\}$ which will then be used to construct an encoding $\tilde{X} D_{\bar{r}} G'^T$, where $D_{\bar{r}} = \mathbf{diag}(\bar{g}_{\bar{r}})$ is a diagonal matrix with uniformly random elements in the diagonal. The idea is that the sampler can use rows of $G\tilde{X}$ and columns of another (essentially redundant) encoding $\tilde{X} D_{\bar{r}} G'^T$ to both verify that the columns of $\tilde{X}$ were correctly encoded via G and that the rows of $\tilde{X}$ were also correctly encoded via G', with some extra randomness. (We note that, in practical data availability schemes such as [ASB18, But20], this encoding is already done, minus the additional random multiplier, so this construction incurs only a very small amount of additional work for the encoder.) This can then be used to prove that the complete encoding $G\tilde{X} D_{\bar{r}} G'^T$ was also correctly performed with essentially no additional overhead for the encoder, relative to performing the standard tensor encoding.

Description. We outline the steps of the encoding algorithm below.

1. Commit to the rows of $X = G\tilde{X}$
2. Receive uniformly sampled $\bar{r} \in \{1, \dots, \bar{m}\}$ (*i.e.*, receive uniformly random $\bar{g}_{\bar{r}} \in \mathbf{F}^{n'}$)
3. Construct diagonal with given uniformly random entries, $D_{\bar{r}} = \mathbf{diag}(\bar{g}_{\bar{r}})$
4. Commit to the columns of $Y = \tilde{X} D_{\bar{r}} G'^T$
5. Commit to complete encoding $Z = G\tilde{X} D_{\bar{r}} G'^T = GY$

Discussion. As the randomness in our protocol is public coin, we note that the randomness of step 2 can be either interactive or utilize the Fiat–Shamir transformation. Additionally, note that the step $Y = \tilde{X} D_{\bar{r}} G'^T$ is equivalent to scaling each column of $\tilde{X}$ by its corresponding element in $\bar{g}_{\bar{r}}$, encoding the rows of this scaled $\tilde{X}$ via G', and placing each encoded row as a column in Y. The only overhead over a standard data availability scheme such as the one in [ASB18] comes from scaling the columns, which can be done in linear time in the number of elements in $\tilde{X}$, which is much faster than the encoding time for most practical codes. Additionally, for the remaining sections, we note that the columns of Y and entries of Z depend implicitly on the randomness $\bar{r}$, which is publicly known or otherwise derivable from public knowledge, such as the commitment to the rows of X of step 1.

3.1.2 Sampling algorithm

In the interactive phase, the sampler simply samples a few rows of the otherwise opaque matrix X and a few columns of Y and checks that these encodings are consistent. If so, then the samples received must also be consistent with the encoding. This means that X and Y must be close to the correct column and row encodings, respectively, of some unique matrix $\tilde{X}$. These can be checked against the complete encoding matrix Z to verify that, indeed, the sampled entries of Z are also symbols from the encoding of $\tilde{X}$, as required. The important point to note is that the only work now required from the encoder is simply to open rows of X, columns of Y, and entries of Z, with no additional computation necessary. It is important to note that the sampling algorithm can construct $D_{\bar{r}}$ from $\bar{r}$, which we assume is public knowledge. (For example, if $\bar{r}$ is defined via Fiat–Shamir, then the commitment to the rows of X suffices.)

Description. We outline the steps of the interactive phase below.

1. Sample $S \subseteq \{1, \dots, m\}$ of some (fixed) size $|S|$
2. Sample $S' \subseteq \{1, \dots, m'\}$ of some (fixed) size $|S'|$
3. Receive the S rows of X, written X_S, from the encoder
4. Receive the S' columns of Y, call them y_j for every $j \in S'$, from the encoder
5. Check $X_S D_{\bar{r}} \tilde{g}'_j = (G y_j)_S$ for every $j \in S'$
6. Verify that $Z_{ij} = g_i^T y_j$ for each $i \in S$ and $j \in S'$

If the encoder does not provide a valid response in any one of these steps, the sampler does not accept. If any equality check fails, the data needed to verify the check comprises a succinct proof that the encoding was done incorrectly.

Discussion. Roughly speaking, the second-to-last step performs the standard Ligero-like check described previously in §2.2. This convinces the sampler of three things. First, that the matrix X is indeed q-close to the encoding of some unique (!) matrix $\tilde{X}$, via G. Second, the S rows received from X are correct encodings of the columns of $\tilde{X}$. And, third, that y_j is a collection of symbols from a correct encoding of the rows of $\tilde{X}$. The final check in the last step simply convinces the sampler that the received samples of the matrix Z are symbols from the tensor encoding of $\tilde{X}$. We note that these are all probabilistic statements and we outline the exact probabilities in §3.2.

Guarantees under additional assumptions. One basic assumption is to assume that the agent running the sampling algorithm is also connected to some other (honest) nodes who are also running this sampling algorithm. (We say a node is *honest* here, if, upon encountering a failing check, the node relays a succinct proof that this check was done incorrectly to all of its connected peers.) In this case, we may assume that the number of rows $|S|$ or columns $|S'|$ that were sampled includes not just those of the agent, but also the minimum amount of samples from the other honest nodes in the network. (Such a number may be a parameter set by the protocol, for example.) Note that the agent running the sampling algorithm may also be connected to other nodes which are dishonest—we make no assumptions about how many such dishonest nodes there are, only about the honest ones.

3.1.3 Decoding algorithm

The decoding algorithm attempts to recover the data $\tilde{X}$ from enough samples of the rows of X or the columns of Y. The reconstruction algorithm only succeeds with high probability (see §3.2.4) and assumes that the sampling algorithm §3.1.2 was run before the reconstruction algorithm. From before, we set d and d' to be the distances of G and G', respectively. As a starting point, we assume that the decoder has received at least one of (a) $|S| > m - d$ rows of X or (b) $|S'| > m' - d'$ columns of Y. These sets S and S' should include sampled columns from the sampling algorithm, run before the decoding algorithm. With this, the decoding algorithm is very simple:

1. Verify that $X_S D_{\bar{r}} g'_j = G_S y_j$ for each $j \in S'$
2. If $|S| > m - d$, decode $G_S \tilde{X} = X_S$ via any erasure decoding algorithm for G
3. If $|S'| > m' - d'$, decode $G'_{S'} \tilde{X}^T = (Y^T)_{S'}$ via any erasure decoding algorithm for G'

If any step fails, the algorithm rejects immediately and has a succinct proof that the encoding was done incorrectly. (Steps 2 and 3 do not fail with high probability, assuming step 1 passes.)

Discussion. At a high level, the reconstruction algorithm works since, if we know that the received rows X_S corresponds to a (subset) of a correct encoding of $\tilde{X}$, then G_S is injective for $|S| > m - d$ and hence has an inverse for elements in its range; *i.e.*, we know that $G_S \tilde{X} = X_S$ has a solution over $\tilde{X}$. A similar thing is true over $G'_{S'}$ if $|S'| > m' - d'$. The fact that the received rows S or columns S' are symbols of the encoding of the unique (but unknown) message $\tilde{X}$, given the check performed in step 1, is shown in §3.2.4. In fact, we will show that this is true even in the extreme adversarial case where the rows S or columns S' provided to the reconstruction algorithm are completely adversarially chosen even after knowing the randomness used in the sampling algorithm. The main challenge with this adversarial choice is ensuring that the received rows and columns do, indeed, map to an encoding of some (unique) message, which is guaranteed since we assume that the sampling algorithm has been run prior to decoding, and that this check has succeeded. In lieu of sampling, we note that it suffices also to receive $|S| > m - q$ rows and a single column $|S'| \geq 1$, which pass step 2's check, since the proposition in the subspace distance check's probabilistic implication (5) holds.

3.2 Guarantees

The protocol (viewed as a collection of the above three algorithms) gives the following guarantees. First, it is complete: if the encoder is honest, the sampler will never reject. (And, in turn, the decoding algorithm always succeeds in decoding the provided data.) Second, the sampling algorithm guarantees unique decoding: if a sampler accepts, X is uniquely decodable to some $\tilde{X}$ except with error probability no more than $p + p'$. Finally, the sampling algorithm guarantees correct encoding: if the checks pass, the received rows of X, columns of Y, and sampled elements of Z do indeed correspond to symbols of a correct encoding of $\tilde{X}$, again with error probability no more than $p + p'$. Finally, and most importantly, the decoding algorithm satisfies a very strong notion which we call 'adversarial reconstruction': if the sampling algorithm has been run once before the decoding algorithm, then it only suffices to receive rows of X and/or columns of Y to both verify that these associated rows/columns were correct and to decode $\tilde{X}$ using any erasure decoder.

3.2.1 Completeness

The *completeness* of the protocol; *i.e.*, the fact that if X, Y, and Z are as defined in the commitment phase of §3.1.1, then no sampler fails, is essentially immediate from the definitions of $X = G\tilde{X}$, $Y = XD_{\bar{r}}G'^T$, and $Z = G\tilde{X}D_{\bar{r}}G'^T$. We encourage the reader to read through the protocol, plugging in the definitions of X, Y, Z, and $D_{\bar{r}} = \mathbf{diag}(\bar{g}_{\bar{r}})$, assuming an honest encoder, to understand what is going on in the individual checks.

3.2.2 Unique decoding

The *unique decoding* property states that, after a single run of the sampling algorithm, it must be true that, with error probability no more than $p + p'$, the matrix X must be 'close to' to a unique encoding of some piece of data $\tilde{X}$, in a way we make clear next. Here, as before, we define p and p' as the error probabilities of the sparsity check (4) and the subspace distance check (5) (which depends on the code G used), respectively, where $|S|$ is the number of rows sampled. This has the immediate implication that, if the sampler can recover the underlying data, it indeed recovers the unique data from which the query responses are derived. For this particular proof, we will assume that the matrix G' contains no zero entries to simplify the proofs and the bounds. (This is the case with, *e.g.*, Reed–Solomon, which we use in our concrete instance of the system.) We note that this can be relatively easily extended to the more general case, with some hit to the error probability, if needed, but we do not use this more general case in any practical instance.

Proof. The unique decoding property for this protocol follows nearly immediately from the subspace distance checks performed in the interactive query phase, with one additional observation: for every vector $x \in \mathbf{F}^{n'}$ with no nonzero entries, $D_{\bar{r}}x$ has uniformly distributed entries over the randomness $\bar{r}$.

Now, from the sparsity check (3), we have, for any one $j \in S'$,

$$(XD_{\bar{r}}g'_j)_S = (Gy_j)_S \quad \underset{p}{\Longrightarrow} \quad \|XD_{\bar{r}}g'_j - Gy_j\| < q,$$

where $S \subseteq \{1, \dots, m\}$ is a uniformly chosen random subset of (fixed) size $|S|$ with

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

Since $D_{\bar{r}}g'_j$ is uniformly randomly distributed for any j, then, from the subspace distance check (5), there exists some matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$ such that

$$\|XD_{\bar{r}}g'_j - Gy_j\| < q, \text{ for any one } j \in S' \quad \underset{p'}{\Longrightarrow} \quad \|X - G\tilde{X}\| < q, \tag{8}$$

where p' is the probability of error defined in (6) or (7), depending on the code G. Here, the randomness is over $\bar{r} \in \{1, \dots, \bar{m}\}$. So, by chaining implications, we see that check 3 guarantees with error probability no more than $p + p'$ that the matrix X is q-close to the encoding of a matrix $\tilde{X}$. Written out as a probabilistic implication, this says

$$(XD_{\bar{r}}g'_j)_S = (Gy_j)_S, \text{ for any one } j \in S' \quad \underset{p+p'}{\Longrightarrow} \quad \|X - G\tilde{X}\| < q,$$

where the probability is over $\bar{r}$ and the uniformly sampled subset $S \subseteq \{1, \dots, m\}$ of size $|S|$.

Finally, let $\tilde{X}'$ be any other possible data matrix. Then,

$$\|G(\tilde{X} - \tilde{X}')\| \leq \|X - G\tilde{X}\| + \|X - G\tilde{X}'\| < 2q \leq d,$$

where we have used the definition that $q \leq d/2$ from the discussion in §2.2 and the fact that the distance of G is d. Hence, if the conclusion that $\|X - G\tilde{X}\| < q$ is true (*i.e.*, the check does not erroneously pass, which happens with probability no more than $p + p'$) then $\tilde{X}' = \tilde{X}$ so $\tilde{X}$ is unique.

Discussion. In a certain sense, the proof above convinces any sampler that the matrix X, committed to by the encoder, is q-close to an encoding of some (unknown) matrix $\tilde{X}$. The proof has an obvious place where it is loose: in our construction, we simply 'throw away' any additional certainty that we get from (8) by only 'using' one index $j \in S'$ to prove the claim. (Indeed, in our analysis, the sampler only really needs to sample one random column to achieve the same error probability, so it suffices to set $|S'| = 1$ with no additional error gained.) An important and interesting future research direction would be to quantify how much the probability of error drops as more queries are used in the probabilistic implication.

3.2.3 Correct encoding

The correct encoding property implies that the samples received by the sampling algorithm correspond, with error probability no more than $p + p'$, to an encoding of the underlying data $\tilde{X}$. In fact, the protocol actually guarantees a much stronger statement, which we show next in §3.2.4, but these proofs will serve as a warm-up for this stronger case.

Column proof. We first show that the received columns y_j are correctly encoded by G'. This is fairly easy to see. From the unique decoding argument, the sampler knows that with error probability no more than p,

$$\|XD_{\bar{r}}g'_j - Gy_j\| < q$$

for each $j \in S'$, and, with additional error probability no more than p',

$$\|X - G\tilde{X}\| < q.$$

So, using (2), we have that, for each $j \in S'$,

$$\|G(\tilde{X}D_{\bar{r}}g'_j - y_j)\| \leq \|X - G\tilde{X}\| + \|XD_{\bar{r}}g'_j - Gy_j\| < 2q \leq d, \tag{9}$$

hence $\tilde{X}D_{\bar{r}}g'_j = y_j$ since G has distance d, which proves the claim that the received columns y_j are correct encodings of the rows of $\tilde{X}$ via $G'D_{\bar{r}}$, except with probability no more than $p + p'$.

Row proof. We now show that the received rows X_S are correctly encoded by G. This proof is slightly more involved. Note that the sampler has checked that

$$X_S D_{\bar{r}}g'_j = G_S y_j,$$

for each $j \in S'$ (though only one j suffices) and also knows that $\|X - G\tilde{X}\| \leq q$, with error probability no more than $p + p'$. So, from the conclusion of the previous proof, since

$$y_j = \tilde{X}D_{\bar{r}}g'_j,$$

for each $j \in S'$, we know

$$X_S D_{\bar{r}}g'_j = G_S\tilde{X}D_{\bar{r}}g'_j,$$

or, equivalently, that

$$(X_S - G_S\tilde{X})D_{\bar{r}}g'_j = 0.$$

Since $D_{\bar{r}}g'_j$ is uniformly distributed over the randomness $\bar{r}$, then, from the zero check, we know that

$$(X_S - G_S\tilde{X})D_{\bar{r}}g'_j = 0 \quad \underset{1/|\mathbf{F}|}{\implies} \quad X_S = G_S\tilde{X}.$$

Putting all of the probabilistic implications together then means that

$$X_S = G_S\tilde{X},$$

except with probability no more than $p + p' + 1/|\mathbf{F}|$.

3.2.4 Adversarial reconstruction

The adversarial reconstruction property we prove below can be interpreted as roughly the following: anyone who runs the sampling algorithm and accepts can receive a number of untrusted queries and responses. These can then be used to decode the underlying data matrix $\tilde{X}$. The reason that these received passing queries and responses are not necessarily trusted is that we make no assumptions about which columns or rows are sent or received. An adversary could then strategically 'pick' rows or columns that are not correctly encoded, but might pass the checks done by the decoder, as the adversary has the freedom to choose any row or column available in X or Y, after observing the sampling randomness.

Result. The result for this particular protocol will be the following: let d be the distance of G and d' be the distance of G', then it suffices for a sampler to run the sampling protocol, with p and p' as defined above, and have either $|S| > m - d$ rows of X or $|S'| > m' - d'$ columns of Y. (Note again that these rows or columns could be adversarially chosen by the encoder, except for those sampled by the node, after observing all randomness in the protocol, including the rows/columns requested by the sampling algorithm.) The sampler will then be able to reconstruct $\tilde{X}$ using the S rows except with probability no more than

$$p + p' + \frac{m}{|\mathbf{F}|} \tag{10}$$

or the S' columns, except with probability no more than

$$m'p + p',$$

where, as before, p is the error probability of the sparsity check (4) and p' is the error probability of the subspace distance check (5) for the code G.

Row proof. We prove the 'row' side first. In particular, we will show that it suffices to receive $|S| > m - d$ rows of X in order to reconstruct $\tilde{X}$ with high probability. To do this, we will show that the sampled rows S are correct encodings of $\tilde{X}$, *i.e.*, with high probability

$$G_S \tilde{X} = X_S,$$

even if S was chosen after observing the sampling randomness. Since the matrix G_S has a left inverse, then running Gaussian elimination suffices to recover $\tilde{X}$. (This is true since G_S has distance at least 1 as we have removed no more than $m - |S| < m - (m - d) \leq d$ rows from G, so G_S is therefore injective.) By assumption, the decoder has run the sampling protocol and therefore verified that $\|XD_{\bar{r}}g'_j - Gy_j\| \leq q$ for some $j \in S'$, fixed in what follows, with error probability no more than p. This, in turn, means that the decoder has therefore also verified that $\|X - G\tilde{X}\| \leq q$, with additional error probability no more than p', over randomness $\bar{r}$. For convenience, we will now set $v_{\bar{r}} = D_{\bar{r}}g_j$, which we note is uniform over $\bar{r}$ in what follows. This means that, using the same reasoning, again, as (9):

$$\|G(\tilde{X}v_{\bar{r}} - y_j)\| \leq \|X - G\tilde{X}\| + \|Xv_{\bar{r}} - Gy_j\| < 2q \leq d, \tag{11}$$

so $\tilde{X}v_{\bar{r}} = y_j$ since G has distance d. From the protocol, the decoder then checks that all rows S satisfy

$$X_S v_{\bar{r}} = G_S y_j$$

for this j. If this is not true, the decoder terminates and has a succinct proof that the encoding was incorrectly performed. (Technically, the decoder only needs to terminate if there are no more than $m - d$ indices of S which pass the test, since otherwise it can still decode $\tilde{X}$. Either way, it now holds a succinct proof that an encoding was incorrectly performed if any index fails.) Finally, since we know that $\tilde{X}v_{\bar{r}} = y_j$ from (11), then

$$X_S v_{\bar{r}} = G_S \tilde{X} v_{\bar{r}}.$$

From the adversarial zero check in appendix A and the fact that $v_{\bar{r}}$ is uniformly random, we then know that

$$X_S v_{\bar{r}} = G_S \tilde{X} v_{\bar{r}} \quad \underset{m/|\mathbf{F}|}{\Longrightarrow} \quad X_S = G_S \tilde{X}.$$

Chaining the probabilistic implications together means that $X_S = G_S\tilde{X}$, and hence the decoder can recover $\tilde{X}$, except with probability no more than

$$p + p' + \frac{m}{|\mathbf{F}|}.$$

Column proof. From before, we would like to prove that, given $|S'| > m' - d'$ columns of Y, call each y_j for $j \in S'$, the decoder can reconstruct $\tilde{X}$ except with very low error probability. Similar to the previous proof, we will show that these received columns are correctly encoded with very high probability and hence we have

$$G'_{S'}\tilde{X}^T = (Y^T)_{S'}.$$

Note that, again, the matrix $G'_{S'}$ is injective since $|S'| > m' - d'$ so we may recover $\tilde{X}^T$ via Gaussian elimination. Now, assume that the S known rows of X are those uniformly sampled by the previously-run sampling algorithm. (If the decoder receives other rows, ignore these.) The decoder then checks that

$$X_S D_{\bar{r}} g'_j = G_S y_j,$$

for each $j \in S'$. If this check fails, the decoder again terminates and has a succinct proof of an incorrect encoding. (As before, this is not technically necessary, unless no more than $m' - d'$ columns pass the check.) On the other hand, if this check succeeds, we know, via the adversarial sparsity check in appendix A, that

$$(Xg'_j - Gy_j)_S = 0, \text{ for each } j \in S' \quad \underset{m'p}{\Longrightarrow} \quad \|Xg'_j - Gy_j\| < q, \text{ for each } j \in S',$$

where the adversarial part comes from the fact that the $j \in S'$ are chosen after the query set S is known. Once this is the case, via the same inequality argument as (11), we have that

$$\|G(\tilde{X}g'_j - y_j)\| < d,$$

for each $j \in S'$, hence

$$\tilde{X} g_j' = y_j,$$

so the received row encodings are correct with error probability no more than

$$m'p + p',$$

as required.

Discussion. In this case, we make no assumptions about the code. In some cases, for example, when G is a Reed–Solomon code over a multiplicative subgroup of order m or the field $\mathbf{F}$ is binary [LAHC16], it may be possible to much more quickly reconstruct the data $\tilde{X}$ than suggested above via Gaussian elimination. Additionally, we note that we have only made worst-case assumptions about the received data: *i.e.*, that the adversary is in complete control of all samples sent to the decoder. In spite of this, the guarantees still hold with high probability, so long as the sampling algorithm has performed a few queries ahead of time.

No sampling. As mentioned before, in lieu of sampling, we note that it suffices also to receive $|S| > m - q$ rows and a single column $|S'| \geq 1$, which pass step 2's check, since the proposition in the subspace distance check's probabilistic implication (5) holds. In this case the error probability is

$$p' + \frac{m}{|\mathbf{F}|},$$

which follows from a nearly identical argument to that of the row proof removing the sampling error p coming from sampling the $|S|$ rows.

3.3 Discussion

There are a number of interesting and potentially useful consequences of this construction. We discuss some below.

Data availability. Data availability, roughly, corresponds to the belief that a piece of data (in our case, $\tilde{X}$) is recoverable from a (potentially adversarial) network. Data availability sampling is a type of protocol which convinces nodes that are connected to a potentially lossy network that a piece of data $\tilde{X}$ can be recovered, under some assumptions, *e.g.*, about the minimal number of honest nodes running the protocol and connected to the network. For example, in the previous construction, we have shown that any node that has run the sampling scheme knows that any further received rows of X or columns of Y can be verified to be correct symbols from the encoding of $\tilde{X}$, with high probability. If this node also believes that there are 'enough' honest samplers also performing this check and accepting, then, in a sense, it can be fairly certain that the data is available. That is: since enough honest samplers have downloaded enough correctly-encoded pieces of data, each of these (a priori unknown to be honest) samples could, in principle, be checked and used to decode

$\tilde{X}$, implying that this data is available. (Dishonest nodes have no effect here, since their data can simply be discarded if the corresponding checks run by the decoder do not pass.) There is at least one set of definitions for 'reasonable' data availability schemes, originally presented in [HSW23] and we show in appendix E that, as a consequence of unique decoding and adversarial reconstruction proved above, our protocol also satisfies these definitions in some settings.

Systematic encoding. In general, it is often useful (in practice) to have G and G' be systematic encodings; this in turn, means that the encoder only has to send X (which will contain $\tilde{X}$ by definition), the columns of Y which do not contain $\tilde{X}$ (as it is already present in X) and the quadrant of Z which does not contain either X nor Y. While we make no assumptions about G, we have one assumption about G': namely that is has no nonzero entries. If G' is systematic, *i.e.*, it can be written as

$$G' = \begin{bmatrix} I \\ P' \end{bmatrix},$$

where I is the identity matrix and P' is another matrix. Note that this clearly has some zeros in its top half, so the assumption made in the original proofs is not satisfied. On the other hand, for Reed–Solomon codes, it is not hard to show that the 'standard' systematic encoding will have P' with no entries equal to zero; we show this in appendix B. In this case, the sampler needs to take at least one column in the nonsystematic part of Y, and the claim is again satisfied. (We, again, make no assumptions about G in our proof, so sampling over rows remains unchanged.)

Distributed encoding. Another observation is that the encoding can be performed in a semi-distributed way: to construct X from $\tilde{X}$, each column needs to be encoded via G, which is an embarrassingly parallel task. Once the column encoding is performed, it is possible to send individual rows to different nodes, who then use those commitments to construct a Merkle tree (which we note has only n leaves, square root of the number of entries in $\tilde{X}$ which, assuming it is square, is n^2). Similarly, anyone who holds a complete row of X, which was necessary to construct the Merkle commitment over the rows, can then encode this row to get a row of Y and Z (after deriving the appropriate randomness from the commitment). A commitment to the rows of Y and entries of Z may be similarly parallelized over a group of nodes who share the appropriate data.

4 Numerics

In this section, we describe a concrete instance of the protocol and show both asymptotics and concrete values in some cases where the parameters are reasonable. We then show that ZODA has the lowest total network communication over all modern protocols (including KZG, and the interactive variation of FRIDA of [EMA24]).

Codes. In this case we set the codes $G \in \mathbf{F}^{m \times n}$ and $G' \in \mathbf{F}^{m' \times n'}$ to be Reed–Solomon codes $G = G'$ that are efficiently encodable over some subset of $\mathbf{F}$, such that Gx can be evaluated in $\sim m \log(m)$ time for any $x \in \mathbf{F}^n$. (The logarithm here is usually taken to base 2.) We set the rate ρ of the code to be $\rho = 1/2$, so $m = 2n$. Note that, in this case, the total matrix size of the data $\tilde{X} \in \mathbf{F}^{n \times n}$ is n^2 field elements, which means that rows and columns are n elements each, and hence much smaller than the size of the total matrix as n grows large.

Error probabilities. Since G is a Reed–Solomon code matrix, the error probability of the subspace distance check is no more than p_{rs}' as given in (7):

$$p'_{\mathrm{rs}} \le \frac{2(d-1)}{|\mathbf{F}|} = \frac{2n}{|\mathbf{F}|},$$

where we have (again) used the fact that $d = m - n + 1 = n + 1$ since Reed–Solomon is maximum distance separable. We will also set $q = n/2 < d/2$ such that, for error probability p, if we want to have

$$\left(1 - \frac{q}{m}\right)^{|S|} \le \left(1 - \frac{(n/2)}{m}\right)^{|S|} = \left(\frac{3}{4}\right)^{|S|} \le p,$$

we need to sample

$$|S| = \left\lceil \frac{\log(p)}{\log(3/4)} \right\rceil$$

rows of X, where the logarithm can be to any base.

Encoding work. With the above parameters, the total amount of work the encoder has to do is (1) encode the columns of $\tilde{X}$ by computing $X = G\tilde{X}$, which can be evaluated in $\sim nm\log(m) \sim n^2 \log(n)$ time (one matrix-vector multiplication of G by each of the n columns), and commit to the rows of X. If the commitment is a Merkle commitment, which we assume in what follows, this takes roughly $m\log(m) \ll mn\log(n)$ time. Then (2) the encoder has to sample randomness $\bar{r}$ from $\{1, \dots, \bar{m}\}$ and construct $D_{\bar{r}} = \mathbf{diag}(\bar{g}_{\bar{r}})$; *i.e.*, construct a uniformly random n-vector $\bar{g}_{\bar{r}} \in \mathbf{F}^n$ using $\bar{r}$ and then set the diagonal elements of $D_{\bar{r}}$ to the entries of $\bar{g}_{\bar{r}}$. After this, the encoder then computes $\tilde{X}D_{\bar{r}}$ by scaling the n columns of $\tilde{X}$ (taking n^2 time; *i.e.*, the number of entries of X) and then computes the encoding of the rows of $\tilde{X}$; *i.e.*, it computes $Y = \tilde{X}D_{\bar{r}}G'^T$, which can also be done in $nm\log(m)$ time. The encoder then commits to these m columns of Y, which also takes negligible time over the encoding. Finally, the encoder computes $Z = GY$ by encoding the columns of Y, which takes $m^2\log(m)$ time and commits to each of its m^2 entries, which takes also $m^2\log(m)$ time, making this the most time-consuming step. This gives a total of around

$$m^2 \log(m) \sim n^2 \log(n)$$

time to run the encoding algorithm, ignoring constants.

Communication total. For now, we allow (again) the matrices $G \in \mathbf{F}^{m\times n}$ and $G' \in \mathbf{F}^{m'\times n'}$ to be arbitrary, since the total communication may be of interest for more general codes than the concrete construction we present below. (We specialize them to the cases introduced at the beginning of this subsection, next.) To begin, the sampler receives three commitments, each of some size C: one for the rows of X, one for the columns of Y, and one for the entries of Z. It also receives the randomness $\bar{r}$ which consists of $\log_2(\bar{m}) = n\log_2(|\mathbf{F}'|)$ total bits if uniform, or consists of the size C of the commitment over the rows of X if using Fiat–Shamir randomness. It then requests $|S|$ rows of X and $|S'|$ columns of Y (which takes $|S|\log_2(m)$ bits and $|S'|\log_2(m')$ bits, respectively), along and openings for each of these. The opening of a Merkle proof of the rows of X (or columns of Y) requires $C\log_2(m)$ bits ($C\log_2(m')$) for each row (column), which means that the total communication is

$$|S|(C\log_2(m) + n\log_2(|\mathbf{F}|)) + |S'|(C\log_2(m') + n\log_2(|\mathbf{F}|)) + 3C.$$

bits for the $|S|$ rows of X and the $|S'|$ columns of Y. Finally, the sampler requests $|S||S'|$ elements of Z, which take a total of $|S||S'|(C\log_2(mm') + \log_2(|\mathbf{F}'|))$ bits. This gives a total amount of communication, in bits, of

$$\underbrace{C(|S|\log_2(m) + |S'|\log_2(m') + |S||S'|\log_2(mm'))}_{\text{commitment total}} + \underbrace{(|S|n + |S'|n' + |S||S'|)\log_2(|\mathbf{F}|)}_{\text{elements total}}.$$

We note that $|S| \lll n$ for essentially any reasonable p with n moderate, as we will see next.

Concrete communication cost. We can specialize this to the case of a Reed–Solomon code for G and G' with the same relative rates $n/m = n'/m' = \rho$, total data size N, in bits, and $n = n'$. This means that the total data square $\tilde{X}$ has $N/\log_2(|\mathbf{F}|)$ elements, which means that $n = \sqrt{N/\log_2(|\mathbf{F}|)}$. Finally, setting $|S'| = |S|$, gives

$$C|S|(|S|+1)\log_2\left(\frac{N}{\rho^2\log_2(|\mathbf{F}|)}\right) + 2|S|\sqrt{N\log_2(|\mathbf{F}|)} + |S|^2\log_2(|\mathbf{F}|),$$

where

$$|S| = \left\lceil \frac{\log(p)}{\log((1+\rho)/2)} \right\rceil.$$

Setting $p \le 2^{-7}$ and $\rho = 1/2$ means that $|S| = 17$ suffices. Setting $p \le 2^{-20}$ means that $|S| = 51$ suffices, and setting $p \le 2^{-80}$ means that $|S| = 193$ samples is enough. As the amount of data, N, becomes large, we note that the total communication is dominated by receiving the field elements for the $|S|$ rows and columns that were sampled, corresponding to the middle term in the above expression:

$$2|S|\sqrt{N\log_2(|\mathbf{F}|)}.$$

Sampling work. The total sampler work is also relatively simple. For each $j \in S'$, the sampler computes $|S|$ matrix-vector products when computing $X_S D_{\bar{r}} g_j$, each of which cost about n operations, and computing an encoding Gy_j which costs $m \log(n)$ operations. This totals to around $|S'|(|S|n + m\log(n))$ operations. It then checks equality between $X_S D_{\bar{r}} g_j$ and $G_S y_j$ and then verifies that $Z_{ij} = (Gy_j)_i$ for each $i \in S$ and $j \in S'$. (It also verifies all Merkle openings of the sent data, but we do not include this cost here since it is negligible relative to the rest of the operations.) This totals to a cost of around

$$|S'|(|S|n + 2n\log(n))$$

operations for the sampler.

Decoding work. The reconstruction work is rather small: it suffices only to reconstruct a message given an erasure-coded Reed–Solomon codeword with no errors. There are several efficient methods to do this, some over binary fields [Tay24], but in general, the cost is still around $n\log(n)$. In either setting, after receiving at least $n+1$ samples of either rows or columns, the algorithm of §3.1.3 gives that the received rows or columns are correctly encoded with high probability, so it suffices to decode each received row (or column) directly, costing around $n^2\log(n)$ total time.

4.1 Comparisons

We compare the current scheme, which is essentially a drop-in replacement for the proof of encoding initially described in [ASB18] and implemented in the Celestia protocol, with 2D KZG [But20] and the interactive variant of FRIDA [EMA24]. The code used to construct these comparisons can be found in

`https://github.com/bcc-research/zoda-numerics`

High level description. We compare the three schemes, put roughly on equal footing, in the following way. First, we define independent samplers for each scheme, each of which samples down to some (fixed) error probability. We then compute the minimum number of such samplers that are necessary to ensure that, collectively, they are able to reconstruct a block, except with probability no more than 2^{-40}. (To do this, we use some basic approximations, which we describe below.)

Collecting samples. It is relatively easy to compute the probability of gathering fewer than ρm samples from a collection of m elements (assuming sampling with replacement) after t tries, where $0 \le \rho < 1$. We would like to ensure that this probability is smaller than $\bar{p}$, which we set to be 2^{-40} for the numerical experiments. In particular, for any $\ell \le \rho m$, there are

$$\binom{m}{\ell}$$

subsets of size ℓ. Landing in any one particular subset of size ℓ happens with probability $(\ell/m)^t$, which means we must have, summing for each subset size $\ell \leq \rho m$, that

$$\sum_{\ell=1}^{\rho m} \binom{m}{\ell} \left(\frac{\ell}{m}\right)^t \leq \bar{p}. \tag{12}$$

(The left hand side here denotes the probability of independently sampling fewer than ρ proportion of rows, while the right hand side is our desired error probability.) One obvious and reasonably tight approximation for ρ greater than around 1/2 and suitably large m is to note that, for each ℓ in the sum, we have $\ell/m \leq \rho$ and that

$$\sum_{\ell=1}^{\rho m} \binom{m}{\ell} \leq \sum_{\ell=0}^{m} \binom{m}{\ell} \leq 2^m.$$

This can be used to bound the sum as

$$\sum_{\ell=1}^{\rho m} \binom{m}{\ell} \left(\frac{\ell}{m}\right)^t \leq 2^m \rho^t.$$

In other words, choosing t to satisfy

$$2^m \rho^t \leq \bar{p}, \tag{13}$$

or, equivalently, getting $t \geq (m + \log_2(1/\bar{p}))/\log_2(1/\rho)$ samples, suffices.

Comparison. Using the above approximation, we choose ρ to be the proportion of samples needed from the encoding in order to decode the original message. (In general, since we only use Reed–Solomon codes in what follows, this is generally the same as the rate of the code in question.) Samplers then sample $|S|$ elements with replacement (as a decent bound for sampling without replacement) in order to ensure that, if the network withholds some fixed proportion of the data, this withheld proportion is $< \rho$ of the data with probability at most 2^{-80} and can therefore be decoded; *i.e.*, we set $|S|$ such that $\rho^{|S|} \leq 2^{-80}$. (In ZODA, we set the number of samples to be $|S|$ such that we are within unique decoding radius; *i.e.*, we set $((1+\rho)/2)^{|S|} \leq 2^{-80}$ which immediately implies the previous condition.) Finally, we then set the number of samplers to be k such that, with probability at least 2^{-40}, the k samplers, who have all sampled independently, without replacement collectively have at least proportion ρ of the data (at least ρm samples). In other words, using the bound (13), we choose k such that

$$2^m \rho^{k|S|} \leq 2^{-40},$$

or, since $\rho^{|S|} \leq 2^{-80}$,

$$k \geq \frac{m + 40}{80}. \tag{14}$$

Based on this, we then compute the total communication the network performs to provide the $|S|$ samples (entries, rows, or cols, depending on the scheme) to each of the k nodes, along with the necessary commitments. Additionally, we write the resulting ratio of this total communication to the size of the original data N as the *network inefficiency*: the total amount of communication the network has to perform over having simply sent the data to exactly one other party. (This ratio is always at least 1.) In table 1, we compare ZODA, the 2D KZG scheme described in [But20] and [HSW23], and the interactive version of FRIDA described in [EMA24], which has substantially lower network inefficiency than FRIDA as described originally in [HSW24].

Protocol instances. We instantiate all three protocols with variable data sizes N ranging from `32MiB` to `1TiB`. For ZODA, we set $|\mathbf{F}| = 2^{128}$, while for ZODA and 2D KZG, we choose the row encoding to have $\rho = 1/2$ (and similarly with the column encoding) which are standard parameters. In contrast, for FRIDA we choose the rate $\rho = 1/4$ to be roughly comparable. We choose $|S|$ in ZODA such that $p \leq 2^{-80}$ (which, in turn, implies that $\rho^{|S|} \leq 2^{-80}$) while the number of samples s in 2D KZG is chosen such that $\rho^s \leq 2^{-80}$. Since FRIDA needs to guarantee unique decoding, we choose the number of samples s such that $((1+\rho)/2)^s \leq 2^{-80}$. For each protocol, k is then chosen to ensure that there are enough samples to reconstruct except with probability no more than 2^{-40}, as computed in (14). Finally, we note that for 2D KZG, to make a fair comparison, we have two variations. One is the vanilla protocol as described in [But20] (which we will call 'elementwise 2D KZG') where nodes sample $|S|$ individual entries until $(1-(1-\rho)^2)^{|S|} = (3/4)^{|S|} \leq 2^{-80}$ (the higher rate reflects the fact that more samples are needed to ensure reconstruction is possible, when sampling only entries of a tensor code). The second is what we will call the row/column-wise protocol, which allows nodes to sample complete rows and columns until $\rho^{|S|} \leq 2^{-80}$. The latter protocol leads to larger per-node communication but substantially lower network inefficiency.

Results. All results are in table 1 for data sizes ranging from `32MiB` to `1TiB`. We note a few interesting findings. First, that ZODA has the lowest total network inefficiency of all protocols: this is due to the fact that any sampler for ZODA does not need to download any additional overhead over the encoding itself, other than the Merkle openings which are very small relative to the size of each row or column. (Contrast this, for example, to 2D KZG, where every sample must come with an additional group element that acts, roughly, as a proof that its corresponding sample is correctly encoded.) Additionally, there is a fairly clear tradeoff: protocols that are not zero overhead, roughly speaking, must trade the per-node communication with the total overhead coming from the network. This is not the case with zero overhead protocols since, by definition, every sample is 'useful' in reconstructing the original data. (Though not all samples are created equal: it may be the case that 'too small' of a sample, such as, *e.g.*, individual entries of the encoding in the case of ZODA, provide no guarantees for the sampling node.) We do note that, at least in ZODA, depending on which tradeoffs we wish to make, it is possible to decrease the node size with only a small amount of additional overhead (or some other tradeoffs). We discuss such approaches in the following section. Finally, we note that the elementwise KZG construction and interactive

	ZODA			2D KZG Row/Col			2D KZG Elementwise			Interactive FRIDA		
	Node	Net	Inef	Node	Net	Inef	Node	Net	Inef	Node	Net	Inef
32M	8.5M	137M	4.3	12M	271M	8.5	96K	1.6G	51	799K	27G	867
256M	24M	1.0G	4.1	35M	2.0G	8.2	240K	32G	128	1.0M	273G	1091
1G	48M	4.1G	4.1	70M	8.1G	8.2	461K	246G	246	1.1M	1.2T	1219
32G	273M	128G	4.0	394M	258G	8.1	2.5M	42T	1346	1.5M	51T	1634
1T	1.5G	4.0T	4.0	2.2G	8.1T	8.1	13M	7.4P	7567	1.9M	2.1P	2147

Table 1: Table comparing the per-node communication (Node), the total network communication (Net), and the network inefficiency ratio (Inef), for each of the four protocols, for a variety of (unencoded) data sizes listed on the leftmost column. Here, K stands for KiB; similarly, M stands for MiB, and so on.

FRIDA, while having very small node sizes, have very high network inefficiency: since every node needs to either download commitments or additional proofs whose data cannot (at least in any obvious way) be used to reconstruct the original data, a lot of the data is 'pure' overhead. In other words, the proofs or commitments, which make up most of the sampler's communication, serve only to prove that the data is correctly encoded, but cannot otherwise be used for reconstructing the data itself.

5 Extensions

The ZODA protocol is relatively flexible in a number of ways we describe below. For example, it is possible to, *e.g.*, reduce node sizes at the cost of some (small amount of) overhead and increased inefficiency, or use a variety of field sizes, depending on the desired application.

Succinct proofs. A simple alternative, which allows for smaller per-node cost at some potentially increased network inefficiency, is to 'wrap' the ZODA sampler within a general purpose succinct proof. This addition would allow samplers to retain the unique encoding guarantee while sampling far fewer rows or columns. This, in turn, elides the need for a decoder to have additional samples, since the succinct proof can be used to ensure that the unique encoding condition holds directly. We note that this is, in theory, more efficient than proving that the encoding was done correctly via a general-purpose succinct proof, while providing very similar guarantees for samplers and decoders.

Field extension. In the proofs provided for the sampling and decoding algorithms, we only make use of a single column being correctly encoded to receive the stated guarantees. Barring potential future work showing that additional column samples contribute to a lower error probability, it is possible to asymmetrically sample more rows than columns. If this is the case, then we note that the only time that a large field needs to be used is in constructing the random diagonal matrix $D_{\bar{r}}$ to guarantee low error probability. In this case, we can view the matrix $\tilde{X} \in \mathbf{F}^{n \times n'}$ and its encoding $X = G\tilde{X} \in \mathbf{F}^{m \times n'}$ as having elements in some base field $\mathbf{F}$, while $Y = \tilde{X} D_{\bar{r}} G'^T \in \mathbf{F}'^{n \times m'}$ has elements in some field extension $\mathbf{F}' \supseteq \mathbf{F}$. It is then possible to trade off the field sizes to ensure that the error probability remains low and the per-node communication is minimized, subject to this constraint. Note that this does not result in a zero-overhead protocol since the field extension's elements do not carry additional information that can reconstruct the corresponding rows. On the other hand, this construction is still, roughly, a drop-in replacement for the construction of [ASB18] since the tensor code remains essentially unchanged. We explore this ZODA extension in appendix C.

ZODA Hadamard variation. If we do not seek a drop-in replacement for tensor codes, it is possible to significantly improve the per-node communication at the cost of introducing a small amount of additional overhead. In particular, as noted previously, only a single column is used in guaranteeing that the matrix X decodes uniquely to some $\tilde{X}$. In this case, it is possible to simply construct a small number of uniformly random vectors, stacked into the columns of a matrix, G'_R, set $Y = \tilde{X} G'^T_R$, and, essentially, run ZODA's sampler directly. The decoder functions very similarly to the one specified in §3.1.3 but can only reconstruct over the received rows, unless G_R is very large, which leads to impractical encoding times as the original data (and therefore $\tilde{X}$) grows large. This modification also allows the use of smaller field sizes, which results in additional savings, again at the cost of some network inefficiency. Additionally, since the resulting code is not a tensor code, there are fewer ways to decode the result and this decoding likely cannot be as easily distributed as vanilla ZODA, or its field extension variation. We explore this Hadamard variation in appendix D.

Minimal samplers. Finally, we note that, as mentioned in the decoding algorithm, §3.1.3, it suffices to either have sampled rows and columns ahead of time and receive $m - d$ rows, or to receive $m - d/2$ rows (versus $m - d$) and receive a single column. Since this is the case, it is possible that individual samplers, which receive essentially no guarantees themselves, sample a single row and a single column. In doing so, while they individually will get no guarantee about unique encoding, *etc.*, they are able to contribute samples which can later be used to reconstruct the original data without this knowledge. Such a sampler would have at least two orders of magnitude less communication than one which seeks to have the unique encoding guarantee to 2^{-80} or so, while still contributing to network security. We note that, with some additional work, it is similarly possible to decode over the columns if enough columns (such as, say, $m' - d'$ columns) are provided along with a (small number of) rows, but some additional care has to be taken to ensure that the rows are known to be (roughly) uniformly randomly sampled.

Conclusion

We have presented ZODA, a flexible protocol that allows a (tensor) code to function as a proof of its own correct encoding with minimal additional computation and no additional storage. We've shown that ZODA remains sound even under adversarially selected samples if only a small number of random samples are verified ahead of time. ZODA can function as a drop-in replacement for the proof of encoding used in the data availability scheme of [ASB18], which is deployed in the Celestia protocol, with stronger guarantees for the sampler and without the need for redundant re-encoding by full nodes.

Future work. There are several questions that are left open by this work. First, are there other zero-overhead proofs of encoding? In a recent note, [EMA24] show that the overhead of the FRIDA protocol can be securely reduced, resulting in significantly lower network inefficiency for a given node size. However, significant overhead remains as correlated queries in all but the first FRI oracle convey no additional information that is useful for decoding. Second, can additional column samples contribute to soundness in ZODA? A positive result would allow for smaller field sizes, resulting in lower communication requirements for ZODA nodes.

Acknowledgments

The authors would like to deeply thank John Adler, Dev Ojha, Nashqueue, Sanaz Taheri, and Mustafa Al-Bassam for their time and all of the helpful conversations regarding Celestia, data availability sampling, and applications of ZODA to consensus and scaling, which were invaluable in writing this paper. We would also like to thank Tarun Chitra and Kobi Gurkan for their helpful comments and corrections, many of which we have included in this paper.

References

[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.

[BCI$^+$23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed–Solomon Codes. *Journal of the ACM*, page 3614423, August 2023.

[But20] Vitalik Buterin. 2D data availability with Kate commitments, 2020. Accessed: 14 September 2024.

[CT05] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In Pierre Fraigniaud, editor, *Distributed Computing, 19th International Conference, DISC 2005, Cracow, Poland, September 26-29, 2005, Proceedings*, volume 3724 of *Lecture Notes in Computer Science*, pages 503–504. Springer, 2005.

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

[DP24] 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. Sampling for proximity and availability. `https://baincapitalcrypto.com/sampling-for-proximity-and-availability/`, 2024. Accessed: 2024-11-10.

[FLLY24] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Lei Yang. Permissionless verifiable information dispersal (data availability for bitcoin rollups). Cryptology ePrint Archive, Paper 2024/1299, 2024.

[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.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[LAHC16] Sian-Jheng Lin, Tareq Y. Al-Naffouri, Yunghsiang S. Han, and Wei-Ho Chung. Novel polynomial basis with fast fourier transform and its application to Reed–Solomon erasure codes. *IEEE Transactions on Information Theory*, 62(11):6284–6299, 2016.

[NNT22] Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. In Maurice Herlihy and Neha Narula, editors, *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*, pages 180–197. ACM, 2022.

[Rab89] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[Tay24] Christopher Taylor. `Leopard-RS`: MDS Reed-Solomon erasure correction codes for large data in c, 2024.

A Checks

In this appendix, we prove the checks used in the proofs of guarantees for the protocol. We call a check *adversarial* if the adversary is able to pick, from any part of the available data, any subset they wish, after observing the randomness drawn from the distribution.

Zero check. The simplest check we use is the *zero check*. Given a code $G \in \mathbf{F}^{m \times n}$ with distance d and some vector $x \in \mathbf{F}^n$, we have that

$$(Gx)_r = 0 \quad \underset{p}{\implies} \quad x = 0,$$

where r is sampled uniformly from $1, \dots, m$, and $p \leq 1 - d/m$. This is relatively easy to see: if $x \neq 0$ then there are at least d elements which are nonzero (by the definition of distance), hence the probability that we choose at least one of these elements is at least $1 - d/m$. We commonly use the special case that G is the Hadamard matrix such that its rows contain all possible tuples of $\mathbf{F}^n$ which means $m = |\mathbf{F}|^n$. (In this case, choosing a random row r corresponds to choosing a uniformly random vector g_r from $\mathbf{F}^n$.) The distance of this matrix is $d = |\mathbf{F}|^n - |\mathbf{F}|^{n-1}$ so the error probability p is bounded from above by

$$p \leq 1 - \frac{d}{m} = 1 - \frac{|\mathbf{F}|^n - |\mathbf{F}|^{n-1}}{|\mathbf{F}|^n} = \frac{1}{|\mathbf{F}|}.$$

Adversarial zero check. The first adversarial check is the *adversarial zero check*. In this check, we have a collection of k vectors $x_1, \dots, x_k \in \mathbf{F}^n$ and some code $G \in \mathbf{F}^{m \times n}$ with distance d. From the zero check above, we know that, for each $i = 1, \dots, k$:

$$(Gx_i)_r = 0 \quad \underset{p}{\implies} \quad x_i = 0, \tag{15}$$

where r is drawn uniformly from $1, \dots, m$, and $p \leq 1 - d/m$. In the adversarial zero check, the index i is drawn *after* the random variable r is picked, which means, equivalently, that this check must succeeds for any choice of i_r. This is equivalent to saying that

$$(Gx_{i_r})_r = 0 \quad \underset{kp}{\implies} \quad x_{i_r} = 0,$$

for any choice of index i_r. This is easy to see from the zero check (15): this implication fails if, and only if, there exists at least one $i = 1, \dots, k$, such that $(Gx_i)_r = 0$ yet $x_i \neq 0$. By the union bound, this happens with probability no more than kp, which gives the claim. In the special case of the Hadamard code, which we use in this paper, the distance is $d = |\mathbf{F}|^{n-1}$ and $m = |\mathbf{F}|^n$, so we have that $p \leq 1/|\mathbf{F}|$ which gives the result.

Adversarial sparsity check. Similar to the previous check is the *adversarial sparsity check*. The standard sparsity check is given in [EA23, §3.2.1], where we have a list of vectors $x_1, \dots, x_k \in \mathbf{F}^n$ and some parameter $q > 0$. In this case, for any $i = 1, \dots, k$,

$$(x_i)_S = 0 \quad \underset{p}{\Longrightarrow} \quad \|x_i\| < q, \tag{16}$$

where $S \subseteq \{1, \dots, n\}$ is uniformly randomly chosen with fixed size $|S|$, and

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

Similar to the adversarial variation of the previous check, the adversary is free to choose any one of these k vectors *after* observing the randomness S. In a very similar way to the previous, we would like to show that

$$(x_{i_S})_S = 0 \quad \underset{kp}{\Longrightarrow} \quad \|x_{i_S}\| < q.$$

This is also similarly easy to see: the probability that any one check fails for some x_i is $\leq p$ by (16). This means that the probability that there exists any x_i for which the check fails is no more than kp by the union bound, which completes the claim.

B Reed–Solomon has no zero entries outside of the systematic block

One standard systematic encoding of a message $x \in \mathbf{F}^{n'}$ into a codeword $G'x \in \mathbf{F}^{m'}$, where the encoding G' is systematic, is to view the message as a polynomial evaluated at m' points, where the first n' of these evaluations correspond to each symbol in the message. In particular, let $\alpha_1, \dots, \alpha_{m'} \in \mathbf{F}$ be some (fixed) evaluation points, and let $f : \mathbf{F} \to \mathbf{F}$ be the unique degree $n' - 1$ polynomial such that

$$f(\alpha_i) = x_i, \tag{17}$$

for $i = 1, \dots, n'$ (*not* m'). We can write this in terms of the Lagrange basis polynomials, defined

$$\ell_j(z) = \prod_{i=1, i \neq j}^{n'} \frac{z - \alpha_i}{\alpha_j - \alpha_i}. \tag{18}$$

These polynomials have the properties that (a) they are of degree $n' - 1$, (b) that

$$\ell_j(\alpha_i) = \begin{cases} 1 & \alpha_i = \alpha_j \\ 0 & \text{otherwise,} \end{cases}$$

for each $i, j = 1, \dots, n'$, and (c) they are nonzero when evaluated at any point which is not one of $\alpha_1, \dots, \alpha_{n'}$ (and, in particular, nonzero when evaluated at any $\alpha_{n'+1}, \dots, \alpha_{m'}$) which is easy to see from their definition (18), or the fact that any polynomial of degree $n' - 1$ has at most $n' - 1$ zeros. It is then clear that we can write

$$f(z) = \sum_{i=1}^{n'} x_j \ell_j(z),$$

which is a (in fact *the*) degree $n' - 1$ polynomial satisfying the definition (17). Finally, we then set the codeword $y \in \mathbf{F}^{m'}$ derived from a message x with corresponding polynomial f to be

$$y_i = f(\alpha_i),$$

for $i = 1, \dots, m'$. Note, then, that we can write $y = G'x$, where G' is defined as

$$G'_{ij} = \ell_j(\alpha_i),$$

for $i = 1, \dots, m'$ and $j = 1, \dots, n'$. This encoding is systematic as it can be written as

$$G' = \begin{bmatrix} I \\ P' \end{bmatrix}$$

where I is the identity (using the definition of the ℓ_j) and

$$P'_{ij} = \ell_j(\alpha_{i+n'}),$$

for $i = 1, \dots, m' - n'$ and $j = 1, \dots, n'$. From observation (c), the matrix P' contains no entries that are zero, which is what we wished to prove.

C ZODA with field extension

As described in the extensions section, here we discuss the possibility of using ZODA where the data is over some field $\mathbf{F}$ while the randomness is over some field extension $\mathbf{F}' \supseteq \mathbf{F}$. As mentioned in the proof in §3.2.2, to get full soundness for the subspace distance check, it suffices to only sample a single column. If a ZODA decoder is only interested in reconstructing over the rows, it can then sample $|S|$ rows (to desired soundness) and a single column, which would keep both p and p' identical to those given in §3.1. (One might not want to do this for other reasons. For example, it may be of interest to ZODA nodes to also guarantee that unique decoding holds over the columns of Y as another possible way of reconstructing rows of X, in which case we would like that $|S| = |S'|$, as in the numerics presented for ZODA.)

Block size	32MiB	256MiB	1GiB	32GiB	1TiB
Per-node communication	1.24MiB	3.35MiB	6.63MiB	37.1MiB	209MiB
Network communication	75MiB	570MiB	2.2GiB	69.6GiB	2.2TiB
Network inefficiency	2.4	2.23	2.2	2.2	2.2

Table 2: Per-node and total network communication along with network inefficiency, for ZODA with field extensions.

Communication. It is also not hard to show the total communication costs in this setting, which is mostly an exercise in bookkeeping. To begin, the sampler receives three commitments, each of some size C: one for the rows of X, one for the columns of Y, and one for the entries of Z. It also receives the randomness $\bar{r}$ which consists of $\log_2(\bar{m}) = n\log_2(|\mathbf{F}'|)$ total bits if uniform, or consists of the size C of the commitment over the rows of X if using Fiat–Shamir randomness. It then requests $|S|$ rows of X and $|S'|$ columns of Y (which takes $|S|\log_2(m)$ bits and $|S'|\log_2(m')$ bits, respectively), along and openings for each of these. The opening of a Merkle proof of the rows of X (or columns of Y) requires $C\log_2(m)$ bits ($C\log_2(m')$) for each row (column), which means that the total communication is

$$|S|(C\log_2(m) + n\log(|\mathbf{F}|)) + |S'|(C\log_2(m') + n\log_2(|\mathbf{F}'|)) + 3C.$$

bits for the $|S|$ rows of X and the $|S'|$ columns of Y. Finally, the sampler requests $|S||S'|$ elements of Z, which take a total of $|S||S'|(C\log_2(mm') + \log_2(|\mathbf{F}'|))$ bits. This gives a total amount of communication, in bits, of

$$\underbrace{C(|S|\log_2(m) + |S'|\log_2(m') + |S||S'|\log_2(mm'))}_{\text{commitment total}} \\ + \underbrace{|S|n\log_2(|\mathbf{F}|) + |S'|n'\log_2(|\mathbf{F}'|) + |S||S'|\log_2(|\mathbf{F}'|)}_{\text{elements total}}.$$

The total amount of communication is listed in table 2 for $|\mathbf{F}| = 2^8$ and extension $|\mathbf{F}'| = 2^{128}$, with $n = n'$, dimensions $m = 4n$ and $m' = 2n$, and G and G' are Reed–Solomon codes.

Discussion. We note that the figures here are substantially smaller than vanilla ZODA for two reasons: the first is that we are only sampling over rows, which means that the network inefficiency should be roughly cut by half, since only half of the data is being transmitted in the first place. (This is not exactly true since the single transmitted column is somewhat large with respect to the row size, but is close enough to be a good guide.) If the protocol was indeed zero overhead, we would see that the network inefficiency should tend to a factor of ~ 2, instead here it can be shown to be bounded from below by 2.1. Second, we cannot reconstruct using the columns with the minimum number of samplers needed to guarantee reconstruction over the rows. This is relatively easy to see using the bound (13) which will in general have the number of nodes needed (which sample only one column each) be substantially smaller than the total number of columns needed to reconstruct. As in vanilla ZODA, there is nothing preventing other nodes, which do sample both columns and rows for reconstruction, from existing in this set up, at potentially increased network communication.

D Hadamard variation

In this section, we describe a Hadamard variation of the ZODA protocol, which is very similar in spirit to the original Ligero protocol, that has similar guarantees to ZODA. From before, this construction results from applying the observations of [EMA24] to a similar construction to that of [HSW23] and ZODA. This Hadamard variation results in less total communication that ZODA, but is not zero-overhead: the sampled columns are only used in verifying that the row encodings were correct, but have no obvious way of being used to reconstruct the data otherwise. (Indeed, one can prove that they cannot be used to aid in reconstructing $\tilde{X}$, unless there are enough columns to reconstruct all of $\tilde{X}$, which is easily seen from the general solution to the matrix system $G\tilde{X} = X$ and $\tilde{X}G'^T = Y$ over $\tilde{X}$, for given G, G', X, and Y of appropriate dimensions.) Additionally, it cannot be used as a drop-in scheme to many of the data availability schemes used in practice today, since sampling can only be done, roughly speaking, over the rows.

Codes. As before, set $G \in \mathbf{F}^{m \times n}$ to be some linear code (we will not use a second code in this case).

Encoding algorithm. The encoder is very similar to the ZODA encoder. Specifically, it begins with some data $\tilde{X} \in \mathbf{F}^{n \times n'}$, then performs the following steps:

1. Encode $\tilde{X}$ to get $X = G\tilde{X}$
2. Commit to the rows of X
3. Receive some uniformly random $G'_R \in \mathbf{F}^{m' \times n'}$ matrix
4. Compute $Y = \tilde{X}G'^T_R$
5. Commit to the columns of Y

Note that the uniform randomness is public-coin and hence may be derived from Fiat–Shamir randomness using, for example, the commitment to the rows of X. Additionally, note that m' can be very small. (For example, in our experiments, we choose $m' = 2$ which suffices for 80 bits of security.)

Sampling algorithm. Let g'_j be the jth column of G'_R, which is publicly known. The sampler then, as before, performs the following steps:

1. Sample $S \subseteq \{1, \dots, m\}$ of some (fixed) size $|S|$
2. Sample $S' \subseteq \{1, \dots, k\}$ of some (fixed) size $|S'|$
3. Receive the S rows of X, written X_S, from the encoder
4. Receive the S' columns of Y, call them y_j for every $j \in S'$, from the encoder
5. Check $X_S g'_j = (Gy_j)_S$ for every $j \in S'$

Similar to ZODA, the sampler does not accept if either it fails to receive all requested rows or columns, or any of the checks do not pass.

Decoding algorithm. Similar to the previous, we assume that the sampler algorithm has been run before the decoding algorithm has and that the rows and columns received there are present in S and S', respectively. In particular, if the number of rows that the decoding algorithm has received is at least $|S| > m - d$, where d is the distance of G, then the decoder can use any erasure decoding algorithm to reconstruct $\tilde{X}$ with high probability.

1. Verify that $X_S g'_j = G_S y_j$ for each $j \in S'$
2. Decode $G_S \tilde{X} = X_S$ via any erasure decoding algorithm for G

Again, as with ZODA, if the number of rows indices in S which pass is smaller than $m - d$, the decoding algorithm does not accept.

Completeness. Completeness is again clear from plugging in definitions.

Unique decoding. The proof of unique decoding is very similar to the one presented in §3.2.2. We only give the error bound here. Let p' be the error probability of the subspace distance check of (5), which depends on the code G and q and $p = (1 - q/m)^{|S|}$, then the error is bounded by

$$p^{|S'|} + p'.$$

Adversarial decoding. The proof again is very similar to that of adversarial decoding, and the corresponding error is

$$p^{|S'|} + p' + \left(\frac{m}{|\mathbf{F}|}\right)^{|S'|}.$$

Block size	32MB	256MB	1GB	32GB	1TB
Per-node communication	572KiB	1.48MiB	2.91MiB	16.1MiB	91MB
Network communication	110MiB	825MiB	3.2GiB	99GiB	3.1TiB
Network inefficiency	3.4	3.2	3.2	3.1	3.1

Table 3: Per-node and total network communication along with network inefficiency, for ZODA's Hadamard variation.

Total communication cost. Tallying the total communication cost is very similar to that of ZODA and similar tricks, such as the field extension trick used there, can also be used here. We present some basic results in table 3 where G is a Reed–Solomon code of rate 1/4, the field is $|\mathbf{F}| = 4$ and its extension is $|\mathbf{F}'| = 2^{64}$; the sampler sets $|S'| = k = 2$. It is of interest that the network inefficiency for the Hadamard variation of ZODA is lower than that of vanilla ZODA, even though the Hadamard variation is not zero-overhead; this is because, in ZODA, every node samples both rows and columns, while in the Hadamard variation, every nodes samples mostly rows (and, say, two columns). While these sampled columns are pure overhead, as they cannot be used to reconstruct the original data, this asymmetric sampling reduces the network overhead at the cost of only being able to reconstruct over row samples. This, in turn, likely prevents good distributed reconstruction algorithms over the Hadamard variation, in contrast to vanilla ZODA or its field extension variation, where such algorithms can use both columns and rows for reconstruction.

E Data availability sampling scheme

We construct a DAS scheme in the random oracle model (ROM) that satisfies the interface and security properties defined in [HSW23, Def. 1].

Construction. For the sake of convenience, we rewrite the definitions used in this paper in terms of a security parameter λ, to match the definitions of [HSW23], below. When reasonable, we attempt to match the notation of that paper in this appendix to aid the reader.

- Finite field $\mathbf{F}$ such that $|\mathbf{F}| > 2^\lambda$
- Data $\tilde{X} \in \mathbf{F}^{n \times n'}$
- Code $G \in \mathbf{F}^{m \times n}$ with distance d
- Code $G' \in \mathbf{F}^{m' \times n'}$ with distance d'
- Random oracle $\rho : \{0, 1\}^* \to \mathbf{F}^{n'}$
- Parameter $\mathsf{q}_S \in \mathbb{N}$ that denotes the number of sampled rows

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$: Return $\mathsf{ck} = \bot$.
- $\mathsf{Encode}(\mathsf{data}) \to (\pi, \mathsf{com})$: Run the encoding algorithm of §3.1.1, where we let:
  - $\pi_X \in (\mathbf{F}^{n'})^m$ denote the vector where the i-th element is the i-th row of X and rt_X denote its Merkle commitment,
  - $\pi_Y \in (\mathbf{F}^{n})^{m'}$ denote the vector where the i-th element is the i-th column of Y and rt_Y denote its Merkle commitment,
  - $\pi_Z \in \mathbf{F}^{(m \cdot m')}$ denote the vector of individual elements of Z and rt_Z denote its Merkle commitment.

  The randomness required in step 2 of the encoding procedure is obtained from the random oracle: $\bar{g}_{\bar{r}} \leftarrow \rho(\mathsf{rt}_X)$. Return $(\pi = (\pi_X, \pi_Y, \pi_Z), \mathsf{com} = (\mathsf{rt}_X, \mathsf{rt}_Y, \mathsf{rt}_Z))$.
- $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$:
  - $\mathsf{V}_1(\mathsf{com}) \to \mathsf{tran}$: Run steps 1-4 of the sampling algorithm of §3.1.2 with $|S| = \mathsf{q}_S$ and $|S'| = \mathsf{q}_{S'}$ and output the sampled rows of X, columns of Y and elements of Z along with the necessary Merkle opening proofs.
  - $\mathsf{V}_2(\mathsf{com}, \mathsf{tran}) \to b$: Compute $\bar{g}_{\bar{r}} \leftarrow \rho(\mathsf{rt}_X)$, verify all Merkle opening proofs and return $\bot$ if at least one fails. Run steps 5-6 of the sampling algorithm of §3.1.2.
- $\mathsf{Ext}(\mathsf{com}, \{\mathsf{tran}_i\}_{i=1}^{\ell})$: Compute $\bar{g}_{\bar{r}} \leftarrow \rho(\mathsf{rt}_X)$ and run steps 1-3 of the decoding algorithm of §3.1.3.

Figure 1: Adapting the ZODA protocol to the framework of [HSW23]

- Parameter $\mathsf{q}_{S'} \in \mathbb{N}$ that denotes the number of sampled columns
- Threshold of verifiers $T \in \mathbb{N}$

The DAS scheme can then be defined, using the definitions of [HSW23], as the tuple of algorithms $\mathsf{ZODA} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{V}, \mathsf{Ext})$ presented in figure 1. As required by the definition of [HSW23], $(\mathsf{Setup}, \mathsf{V}_2)$ are probabilistic polynomial time algorithms and $(\mathsf{Encode}, \mathsf{V}_1, \mathsf{Ext})$ are deterministic polynomial time algorithms.

Index sampler. We use the notion of an *index sampler* introduced in [HSW23, Def. 12]. Our index sampler can be seen as two parallel executions of the "sampling with replacement" algorithm: one for sampling rows, the other for sampling columns. Let $\nu_{\mathsf{wr}} : \mathbb{N}^4 \to [0,1]$ denote the quality of the "sampling with replacement" sampler computed in (12). Let $\nu : \mathbb{N}^7 \to [0,1]$ denote the quality of the joint-row-and-column sampler. The joint sampler fails to output enough rows and columns simultaneously if and only if both the row-index samplers and the column-index sampler fail to output enough distinct indices. Taking the probability of the intersection of independent events, it holds that:

$$\nu(m, d, \mathsf{q}_S, m', d', \mathsf{q}_{S'}, \ell) = \nu_{\mathsf{wr}}(m - d, m, \mathsf{q}_S, \ell) \cdot \nu_{\mathsf{wr}}(m' - d', m', \mathsf{q}_{S'}, \ell), \tag{19}$$

where

$$\nu_{\mathsf{wr}}(m - d, m, \mathsf{q}_S, \ell) = \sum_{t=1}^{m-d} \binom{m}{t} \left(\frac{t}{m}\right)^{\mathsf{q}_S \ell}.$$

(Note that this is the same quantity as (12), so an identical bound holds.)

E.1 Completeness

By the completeness property of §3.2.1, it follows that running V for some honestly encoded data $\tilde{X}$ will always return 1. Therefore, Ext will only abort and fail to return $\tilde{X}$ if the verifiers jointly sampled fewer than $m - d$ rows of X and $m' - d'$ columns of Y. Let $\ell = \mathsf{poly}(\lambda)$ be the number of verifiers and $\mathsf{NotEnough}$ denote the event in which the verifiers fails to sample enough rows and columns. By definition of the quality of an index sampler,

$$\Pr[\mathsf{NotEnough}] \leq \nu(m, d, \mathsf{q}_S, m', d', \mathsf{q}_{S'}, \ell),$$

where ν is computed as in (19).

E.2 Soundness

Let $\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\ell,\mathsf{ZODA}}(\lambda)$ denote the advantage of an adversary $\mathcal{A}$ against the soundness experiment as defined in [HSW23] run for the protocol ZODA and an integer $\ell = \mathsf{poly}(\lambda)$ such that $\ell > T$. We show that this advantage is negligible in λ.

Proof. To see this, assume that $\ell \geq T$. Recall that soundness as defined in [HSW23], requires that the probability that ℓ executions of $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$ all output 1 and the extractor outputs $\perp$ is negligible.

Let $\mathcal{B}$ denote the adversary that opens Merkle commitments honestly and otherwise behaves identically to $\mathcal{A}$. Let $\mathsf{Adv}^{\mathsf{Merkle}}_{\mathcal{A}}(\lambda)$ denote the probability of that $\mathcal{A}$ breaks the binding property of the Merkle commitment scheme for either if the vectors π_X, π_Y, or π_Z. By the binding property of the Merkle commitment, it holds that

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\ell,\mathsf{ZODA}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{B},\ell,\mathsf{ZODA}}(\lambda) + \mathsf{Adv}^{\mathsf{Merkle}}_{\mathcal{A}}(\lambda).$$

By definition of Ext, the adversary $\mathcal{B}$ wins if and only if one of the following events occurs:

- $\mathsf{CheckFails}$: all ℓ executions of V output 1 and the check performed in step 1 of the decoding algorithm fails.

- $\mathsf{NotEnough}$: all ℓ executions of V output 1 and the verifiers jointly sampled fewer than $m - d$ rows of X and $m' - d'$ columns of Y.

Therefore, taking a union bound over the probability of these events occurring:

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{B},\ell,\mathsf{ZODA}}(\lambda) \leq \Pr[\mathsf{CheckFails}] + \Pr[\mathsf{NotEnough}].$$

The probability $\Pr[\mathsf{NotEnough}]$ is given in (19), so we can now focus on the event $\mathsf{CheckFails}$.

CheckFails event. Consider two cases:

1. $\mathsf{CheckFails} \wedge \mathsf{Close}$: all ℓ executions of V output 1, the decoder's check fails and X is close to a unique encoding of some piece of data $\tilde{X}$
2. $\mathsf{CheckFails} \wedge \overline{\mathsf{Close}}$: all ℓ executions of V output 1, the decoder's check fails and X is not close to a unique encoding of some piece of data $\tilde{X}$.

The probability of $\mathsf{CheckFails}$ is the sum of the probabilities of the above two events. By the unique decoding property §3.2.2, a single verifier's check does not erroneously pass, except with probability $p + p'$, hence at least one check fails except with probability at most

$$\Pr[\mathsf{CheckFails} \wedge \overline{\mathsf{Close}}] \leq p + p'.$$

(This probability can be improved, but suffices for our cases.) Consider now the case $\mathsf{CheckFails} \wedge \mathsf{Close}$. If the union of all transcripts only contains rows and columns that are sampled from the unique close encoding of the underlying data, then by the completeness property of the decoding algorithm (§3.2.1), the check will not fail. Therefore if $\mathsf{CheckFails} \wedge \mathsf{Close}$ occurs, then it must be the case that there exists at least one row (or column) in the union of all transcripts that is not equal to the corresponding row (or column) in the unique closest correct encoding of the data. By the correct encoding property (§3.2.3), we know that each verifier's transcript only contains rows and columns that are sampled from the unique close encoding of the underlying data, except with probability $p + p'$. Therefore, taking a union bound over all samplers,

$$\Pr[\mathsf{CheckFails} \wedge \mathsf{Close}] \leq \ell(p + p')$$

Putting everything together, we have shown that

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\ell,\mathsf{ZODA}}(\lambda) \leq (\ell + 1)(p + p') + \nu(m, d, \mathsf{q}_S, m', d', \mathsf{q}_{S'}, \ell) + \mathsf{Adv}^{\mathsf{Merkle}}_{\mathcal{A}}(\lambda)$$

E.3 Consistency

Consistency is closely related to the adversarial reconstruction property of §3.2.4. Adversarial reconstruction states the following: "if the encoding X is close to a unique encoding under G, then the decoding algorithm (and therefore Ext) will either abort, or output the unique data associated with the closest encoding with probability $p + p' + m/|\mathbf{F}|$ if reconstructing over the rows, or $m'p + p'$ if reconstructing over the columns".

Direct guarantee. Obtaining the guarantee that X is close to a unique encoding under G can be done in multiple ways. The trivial method requires Ext to collect more than $m - q$ rows (where $q < d/2$ if using a Reed–Solomon code for G and $q < d/3$ for general codes) and check that these rows are consistent with the encoding of at least one column. This suffices to reconstruct a unique $\tilde{X}$ for the same reasons stated in the discussion of §3.1.3. We note that this directly satisfies the conditions of consistency, where the error probability is no more than $p + m/|\mathbf{F}|$.

Relaxations. Though it is possible to get a guarantee of the same form as that of [HSW23] without additional overhead, note that we only get this guarantee over the rows. This requires more samples than would be necessary to erasure decode. (We suspect it may be possible to get a similar guarantee given, say $m' - q'$ such columns, but leave this for future work.) On the other hand, there are many reasonable, and practical, relaxations of the definition of consistency which allow us to have even more efficient methods for decoding (or 'extracting', in the parlance of [HSW23]).

One way of doing this is to allow the extractor to assume that at least one transcript is honest. This can be practically achieved in multiple ways: the extractor itself could do its own sampling from the network; the extractor could be given a succinct proof that one of the transcripts is correct; or, perhaps, the extractor could be given a transcript that has verifiable randomness. (The definition and algorithms of [HSW23,HSW24] assume this latter case, essentially, and ensure that every node and every transcript contains this proof, which we note is pure overhead as, in our case, it suffices only to have one.) In any case, our definition of the decoding algorithm, and its adversarial decoding property, ensure that such a 'relaxed' extractor outputs the correct $\tilde{X}$ given enough (potentially adversarially-selected) transcripts with enough rows or columns.