Data Availability for Thousands of Nodes

Yanpei Guo
National University of Singapore
guo.yanpei@u.nus.edu

Alex Luoyuan Xiong
National University of Singapore
alex.xiong.tech@gmail.com

Wenjie Qu
National University of Singapore
wenjiequ@u.nus.edu

Jiaheng Zhang
National University of Singapore
jhzhang@nus.edu.sg

***Abstract*—Scalable data availability (DA) is essential for high-throughput, decentralized blockchains, enabling lightweight nodes to verify block availability without incurring the prohibitive costs of full data replication. Reed-Solomon (RS) code commitment schemes underpin modern DA protocols by ensuring that dispersed data fragments can be verified as part of a valid codeword, even in the presence of malicious block producers. However, state-of-the-art schemes such as FRIDA (Crypto'24), while computationally efficient, incur substantial per-node communication overhead at the scale of thousands of network nodes, often 5.7$\times$ the size of the actual data fragment.**

In this work, we introduce CONDA, a new interleaved RS code commitment scheme that significantly reduces the communication overhead while retaining FRIDA's prover efficiency. At its core is a novel evaluation consolidation technique for polynomial commitment scheme (PCS) that reduces the problem of proving n evaluations at fixed points (one per verifier) to a single evaluation at a random point, using logarithmic communication. This technique is lightweight, hash-based, and compatible with any multilinear PCS.

To further optimize for DA applications, we introduce LightLigero, a new multilinear PCS that improves upon DeepFold (Sec'25) with $O(\log n)$ reduction in proof size and only 30% slowdown in prover time. Combining CONDA and LightLigero yields an efficient DA scheme for thousands of nodes.

Our implementation demonstrates a 4$\times$ reduction in communication cost compared to FRIDA, while incurring only a 25% increase in prover time. It also achieves near-best prover time and near-best communication cost simultaneously among all code commitment schemes. CONDA also offers at least $3\times$ smaller proofs and $4\times$ faster provers than state-of-the-art verifiable secret sharing constructions such as ZXH+22 (Sec'22) and PolyFRIM (Sec'24).

I. Introduction

Public blockchains replicate state across many nodes, tolerating Byzantine faults through consensus. Ensuring full accessibility of the block data is necessary for both safety and liveness. However, as blockchains scale to thousands of nodes [1] and high-throughput blocks with thousands of transactions [2], the full-replication model where every node downloads the entire block becomes prohibitively expensive.

Data availability (DA) protocols, initially developed for distributed storage systems [3], [4] and recently adapted for blockchains [5], [6], manage to ensure data accessibility in the Byzantine fault tolerance setting while avoid incurring linear communication overhead, as in full replication models.

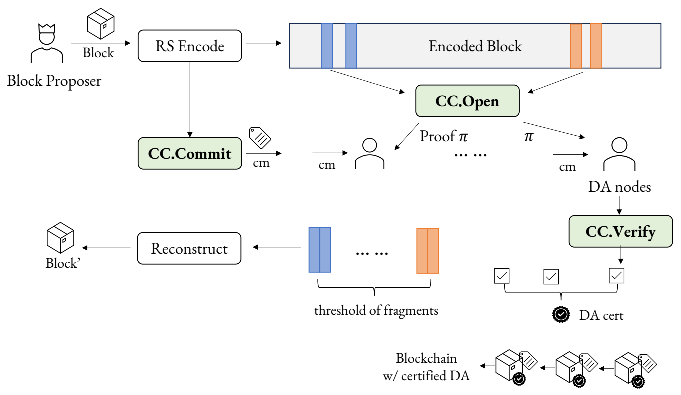

Fig. 1: Data availability workflow

By employing *erasure codes* [7], these protocols allow each node to receive only a small fraction of a block's data, rather than the entire block. Provided that a sufficient number of correct fragments are available, honest nodes can reconstruct the original block. Dispersing fragments across a larger set of nodes improves fault tolerance, mitigating attempts by faulty participants to suppress or corrupt block data. Moreover, increasing the number of DA nodes reduces per-node communication and storage demands, lowering the entry barrier for resource-constrained users, ultimately enhancing the blockchain's decentralization.

Assume that the block content is arranged into a matrix $M \in \mathbb{F}^{L\times k}$. All exiting DA schemes [8], [9], [10], [11] use *interleaved Reed-Solomon* (RS) codes [12] $\mathsf{RS}^{\equiv L}[k,n,\mathbb{F}] : \mathbb{F}^{L\times k} \to (\mathbb{F}^L)^n$ that encode M into a *codeword* consisting of n *symbols*, each being a vector $\in \mathbb{F}^L$, by interpreting each row of M as the coefficients of a degree-$(k-1)$ polynomial $f(X)$ and evaluating it at n fixed points $(e_1, \cdots, e_n) \subset \mathbb{F}^n$. RS code with *rate* $\rho = \frac{k}{n} \in (0,1]$ allows recovery of M from any k symbols. Ideally, the message matrix M is wide such that $n \geq N$, where N is total number of nodes. This ensures that each node can store a fragment of size $\frac{M}{\rho N}$, containing a distinct set of symbols.

Although the description of DA appears straightforward, a major complication arises in blockchain systems: the block proposer, i.e. encoder, may be malicious. In such cases, symbols dispersed to DA nodes may not originate from a valid codeword. As a result, even if honest nodes collect a sufficient number of fragments, they may be unable to reconstruct the original message. To address this issue, all existing DA protocols incorporate a cryptographic primitive called *RS Code Commitment Scheme* (CC) [13]. Intuitively, a code commitment scheme enables a prover to cryptographically commit to a valid codeword while enabling verifiers to open and verify individual symbols from the commitment. When used in DA, the commitment is broadcast to all nodes who will collectively reach consensus on the availability of the committed block. The whole procedure is illustrated in Figure 1.

Hall-Andersen et al. [13] proves that any CC naturally yields a secure DA protocol. Moreover, the efficiency of DA protocol highly depends on the efficiency of CC scheme, which is primarily determined by two factors: the per-node communication overhead (comprising the commitment and opening proofs), and the total prover time to commit the codeword and generate all per-symbol opening proofs. Thus, our work only needs to focus on improving CC schemes on these two factors.

Early CC schemes [8], [14] are constructed using group-based cryptographic primitives. These constructions rely on either *homomorphic* vector [8] or polynomial commitments [14]. In both cases, the prover's workload requires at least $O(|M|)$ group operations over elliptic curves, which is computationally prohibitive for large block size.

To address this overhead, recently, Hall-Andersen et al. [15] introduced another CC called FRIDA (Crypto'24), which leverages *Interactive Oracle Proof of Proximity* (IOPP) [10] for RS codes. FRIDA enables significantly faster provers by proving proximity of the received symbols to a valid codeword, avoiding costly group operations and relying solely on lightweight hash functions. Despite this efficiency gain, a key limitation lies in its concrete communication overhead, which typically exceeds 500KB. This overhead is manageable when the number of nodes N is small, but becomes prohibitive when N approaches the order of thousands as this communication overhead will exceed the data fragment size and dominate per-node communication. Concretely, when dispersing a 64MB block to 2048 nodes[1], the per-node code commitment overhead (the commitment plus opening proofs) is 736KB whereas the actual received data fragment is only 128KB. Such $5.7\times$ overhead is unacceptable and will exacerbate when scaling to large networks such as the Ethereum blockchain which has 6000 detectable validators at the time of writing[2].

Recall that a sufficient number of DA nodes is necessary to ensure the security and decentralization of the blockchain. The inefficiency of FRIDA motivates our research question: *Can the communication overhead of* FRIDA *be significantly reduced in the presence of thousands of nodes while preserving its prover efficiency?*

[1]The full setup is $k = 2^{16}, L = 2^5, \rho = 0.25$, $\mathbb{F}$ is the scalar field of BN254 curve – the 64MB data is encoded into an encoded block of 256MB.

[2]As of April 2025, 6680 validators are active on Ethereum network: https://etherscan.io/nodetracker

A. Our Contributions

We affirmatively answer the above research question, and summarize our contributions as follows:

- CONDA**: Communication-efficient interleaved RS code commitments**: We propose CONDA, a new interleaved RS code commitment scheme based on a novel consolidation technique. Beyond its efficiency, CONDA enables flexible performance tradeoffs as it can be paired with any multilinear polynomial scheme. Using compiler proposed in [13], CONDA can be compiled into an efficient DA protocol for thousands of nodes.
- **Multi-evaluation via evaluation consolidation**: The underlying cryptographic technique of CONDA is a new paradigm for the PCS multi-evaluation: through a novel *evaluation consolidation* protocol that reduces the task of proving n individual evaluations to proving a single evaluation at a random point, convincing all verifiers of the original statements. The prover cost of the consolidation is dominated by $O(M)$ hashes, and the per-verifier communication is $O(\log^2 N/\log\log N)$. Our consolidation protocol is compatible with any multilinear PCS.
- LightLigero**: multilinear PCS based on RS code with $O(\lambda \log n)$-sized proof**: To further optimize the performance of CONDA, we propose LightLigero, a novel multilinear PCS tailored for DA applications. It achieves $O(n \log n \cdot \mathbb{F} + n \cdot \mathbb{H} + \frac{n}{\log n} \cdot \mathbb{G})$ prover time, and $O(\lambda \cdot \log n)$ proof size. By introducing an immaterial $\frac{n}{\log n} \cdot \mathbb{G}$ prover overhead, LightLigero reduces the proof size by a factor of $\log n$ compared to RS code-based PCS [10] whose proof size is $O(\lambda \log^2 n)$. This tradeoff is especially valuable in communication-sensitive DA applications.
- **Implementations and evaluations**: We implemented our scheme in Rust and compared it against state-of-the-art code commitments.[3] Our scheme achieves a $3{\sim}4\times$ reduction on per-node communication from FRIDA (Crypto'24) while retaining its prover efficiency (roughly 25% slowdown) across all parameter regimes. Among all relevant code commitments, ours simultaneously achieves near-best prover speed and near-best communication cost.
Compared with state-of-the-art RS code-based multilinear PCS Deepfold (Sec'25), LightLigero has $3\times$ smaller proof size with only 30% increase in prover time. Our evaluation consolidation technique also benefits verifiable secret sharing (VSS): Relative to ZXH+22 (Sec'22) and PolyFRIM (Sec'24), our construction offers at least $3\times$ improvements in proof size and $4\times$ improvements in prover time.

II. Related Work

A. Data availability

The functionality that enables participants to verify the retrievability of large data blobs without fully downloading them has long been studied. *Proofs of retrievability* (PoR) [16] considers a trusted data writer who correctly encodes the data

[3]Code available at: https://0x0.st/8VOF.zip

TABLE I: Complexities of RS code commitment schemes with message size M and node number N in DA protocols.

Schemes	Per-node Communication	Prover	Verifier
NNT [8]	N	$M \cdot \mathbb{G}$	$(N + \frac{M}{N})\mathbb{G}$
ADVZ [14]	$\frac{M}{N}$	$N \log N \cdot \mathbb{G}_E + M \cdot \mathbb{G}$	$\frac{M}{N} \cdot \mathbb{G}$
FRIDA [15]	$\lambda \cdot \log^2 M$	$M \log M \cdot \mathbb{F} + M \cdot \mathbb{H}$	$\lambda \log^2 M \cdot \mathbb{H}$
ZODA [11]	$\lambda \cdot \sqrt{M}$	$M \log M \cdot \mathbb{F} + M \cdot \mathbb{H}$	$\lambda\sqrt{M} \log M \cdot \mathbb{F} + \lambda \cdot \sqrt{M} \cdot \mathbb{H}$
CONDA + Dory	$\log M + \frac{\log^2 N}{\log\log N}$	$M \cdot \mathbb{G}$	$\log M \cdot \mathbb{G} + \frac{\log^2 N}{\log\log N} \cdot \mathbb{F}$
CONDA + LightLigero	$\lambda \log M + \frac{\log^2 N}{\log\log N}$	$M \log M \cdot \mathbb{F} + M \cdot \mathbb{H} + \frac{M}{\log M} \cdot \mathbb{G}$	$\lambda \log M \cdot \mathbb{H} + \frac{\log^2 N}{\log\log N} \cdot \mathbb{F}$

• Per-node communication only consider commitment and proof, doesn't include data fragment.
• Prover time includes code commit time and generating all proofs for N nodes.
• $\mathbb{G}_E$ denotes group exponentiation, $\mathbb{G}$ denotes group MSM, $\mathbb{H}$ denotes Hash operation, $\mathbb{F}$ denotes field multiplication; Normally, for single operation time, there has $\mathbb{G}_E > \mathbb{G} > \mathbb{H} > \mathbb{F}$.
• For simplicity, the asymptotic notation $O(\cdot)$ is omitted.

before storing them on untrusted servers that provide proofs of possession to verifiers' queries. Extending to a more Byzantine setting, we informally use the umbrella term *data availability* (DA) to refer to the challenge of providing cryptographic retrievability guarantees even in the presence of malicious data encoders and faulty storage nodes. Data availability solutions are mainly used in two ways: for scalable data dispersal during fault-tolerant replications or for post hoc data (un)availability detections. The former is known in the literature as *verifiable information dispersal* (VID) and the latter as *data availability sampling* (DAS). The differences between VID and DAS are subtle and can be safely ignored for our purposes, as both rely on the same technique, namely RS code commitments, which dominates their overall cost.

Verifiable information dispersal. Information dispersal was introduced by Rabin [3] as a specialized application of error-correcting codes, later extended to verifiable protocol in asynchronous networks [17] with subsequent improvements [14], [18], [6], [19] on the dispersal and retrieval cost.

The recent resurgence of interest in VID is driven by its potential to scale blockchain systems. Yang et al. [6] propose a scheme that commits RS-encoded data using a Merkle tree, where each encoded symbol is accompanied by a Merkle proof. Nodes can verify that their received chunks are consistent with the same encoded data, ensuring unanimous recovery or rejection if enough chunks are verified, but they cannot guarantee the chunks originate from a valid codeword, risking failure of retrieving the original data.

The underlying problem of Yang et al. [6] is it only offers vector commitment instead of code commitment. To solve this problem, Nazirkhanova et al. [8] propose a RS code commitment scheme utilizing homomorphic vector commitments (e.g., KZG [20] or Pedersen [21]) to enable strong retrievability of the original payload with a highly efficient disperser. Its encoder has $O(M)$ group MSM and it has commitment of size $O(N)$, which is not practical when N is large.

Alhaddad et al.[14] and Bearer et al.[9] propose a RS code commitment scheme using fast multi-evaluation algorithms of univariate KZG [22], [23]. When the message matrix is arranged into $L \times k$, their commitment size is $O(L)$. Arranging matrix into different shapes can bring different tradeoffs. A significant weakness of their work is its prover cost is dominated by the $O(k \log k)$ group exponentiation when the matrix is quite wide. The most prover-efficient setting is $k = \rho N$.

Data availability sampling. In DAS, a large set of lightweight clients randomly samples small chunks of already committed data to detect potential unavailability. The retrievability property also differs qualitatively between the two approaches. In DAS, data is *probabilistically* available, with increasing confidence with more queried samples until the number of distinct samples reaches the reconstruction threshold, at which point retrievability is guaranteed. In contrast, VID provides a binary assessment of data availability: below the threshold, the verified chunks offer no assurance; above the threshold, retrievability is deterministically ensured under the honest majority assumption.

DAS was first introduced by Bassam et al. [5] to enable light nodes to verify the integrity of ever-growing blockchains without requiring access to entire blocks. Initially, it was proposed as a high-level mechanism without a detailed cryptographic construction.

Later, Hall-Andersen et al. formalized DAS schemes [13] and presented a compiler from any RS code commitment to a secure DAS. They constructed a hash-based interleaved RS commitment using Ligero [24]-inspired techniques at the cost of higher communication cost – with commitment of size $O(\lambda\sqrt{M})$, where λ is the query count of Ligero.

In FRIDA[15], Hall-Andersen et al. extended their earlier work in [13] by introducing a new transparent RS code commitment based on FRI [10] with blazing fast prover. Although the commitment size has been reduced compared with [13], it's still $O(\lambda \log^2 M)$. Concretely, when $M = 2^{20}$, the commitment size is roughly 500KB.

Lately, Evans et al. introduced ZODA [11] that used 2D RS code to construct a new code commitment. Notably, they avoid sending any additional opening proof and their commitments are only three Merkle commitments (96 bytes). Nodes randomly sample a large number of 1D-encoded columns and rows and re-encode them in another direction to check the consistency of their intersected cells, which serves as the proof of correct encoding. Similarly to FRIDA, their per-node communication is dominated by the $\lambda \geq 100$ number of rows/columns sampled to achieve a 2^{-80} soundness error.

TABLE II: Complexities comparison of VSS schemes across N nodes

Scheme	Dealer time	Communication	Transp.
KZG [22]	$N \log N \cdot \mathbb{G}_E$	1	no
ZXH+22 [23]	$N \log N \cdot \mathbb{H}$	$\log^2 N$	yes
PolyFRIM [25]	$N \log N \cdot \mathbb{H}$	$\lambda \log^2 N$	yes
CONDA + Dory	$N \log N \cdot \mathbb{F} + N \cdot \mathbb{G}$	$\frac{\log^2 N}{\log \log N}$	yes

• We omit verifier time because all schemes has identical verifier complexity and proof size;
• $\mathbb{G}_E$ denotes group exponentiation, $\mathbb{G}$ denotes group MSM, $\mathbb{H}$ denotes hash operations, $\mathbb{F}$ denotes field multiplication. Concretely, there has $\mathbb{G}_E > \mathbb{G} > \mathbb{H} > \mathbb{F}$.
• For simplicity, the asymptotic notation $O(\cdot)$ is omitted.

B. Verifiable secrete sharing

Due to the close relationship between Shamir secret sharing [26] and RS code, verifiable secret sharing can be treated as a special case of RS code commitment. The well-established notion of VSS was first introduced by Chor et al. [27]. Similar to DA, a central challenge in VSS lies in enabling the dealer to prove that all dispersed shares are derived from a low-degree polynomial. Kate et al. [20] proposed the first VSS scheme with *succinct* broadcast cost, achieving sublinear communication complexity relative to the number of parties N. Their key insight was to broadcast a compact polynomial commitment and generate proofs for N distinct evaluations or perform a batched multi-evaluation for the commitment.

By leveraging the structure of the FFT to evaluate N points in $O(N \log N)$ time, Tomescu et al. [28] achieved a prover time of $O(N \log N \cdot \mathbb{G}_E)$ for KZG multi-evaluation, and later reduced the proof size to $O(1)$ [29], matching that of the original KZG scheme. However, the concrete prover time remains high due to the cost of group exponentiations, taking over 2800 seconds to generate all proofs for $n = 2^{20}$. Furthermore, it relies on a trusted setup, where the presence of trapdoors can compromise overall security.

Zhang et al. [23] introduced a *one-to-many* zero-knowledge proof system with a batched prover that can generate N proofs for a circuit with N outputs, where each verifier is concerned with only one output. This approach enables an efficient batched prover for the multi-evaluation of a univariate polynomial in FFT circuits. Specifically, a dealer can commit to a polynomial and efficiently prove N distinct evaluations to N verifiers, achieving $O(N \log N)$ prover time and $O(\log^2 N)$ proof size per evaluation. This significantly reduces the prover cost compared to traditional PCS approaches requiring separate proofs for multiple openings. Furthermore, their scheme is *PCS-agnostic*, allowing flexibility in integrating with different PCS. When paired with transparent PCS, the overall VSS scheme is also transparent.

Building on this, Zhang et al. [25] proposed PolyFRIM, a transparent multivariate polynomial commitment scheme optimized for the same *one-to-many* evaluation setting. PolyFRIM achieves similar asymptotic complexity and a concretely faster prover by leveraging multivariate FRI but sacrifices the PCS-agnostic property of [23] due to its reliance on FRI.

Our proposed RS code commitment scheme can also benefit VSS. We present a summarize of asymptotic complexities in Table II.

TABLE III: Complexities comparison of multilinear PCS of size n

Scheme	Prover time	Proof Size
PST [30]	$n \cdot \mathbb{G}$	$\log n$
Dory [31]	$n \cdot \mathbb{G}$	$\log n$
Hyrax [32]	$n \cdot \mathbb{G}$	$\sqrt{n}$
Orion [33]	$n \cdot \mathbb{H}$	$\lambda \log^2 n$
Ligero++ [34]	$n \log n \cdot \mathbb{F} + n \cdot \mathbb{H}$	$\lambda \log^2 n$
Deepfold [35]	$n \log n \cdot \mathbb{F} + n \cdot \mathbb{H}$	$\lambda \log^2 n$
LightLigero	$n \log n \cdot \mathbb{F} + n \cdot \mathbb{H} + \frac{n}{\log n} \cdot \mathbb{G}$	$\lambda \log n$

• We omit verifier time because all schemes has identical verifier complexity and proof size;
• For simplicity, the asymptotic notation $O(\cdot)$ is omitted.

C. Multilinear PCS

Multilinear PCS is an integral component in our construction. In Table III, we summarize the properties of related schemes for a comparison.

PCS can be broadly categorized into two parts: group-based and hash-based. Group-based schemes such as PST [30] and Dory [31] normally have very short proof size, concretely smaller than 2KB. But their prover time is usually not cheap as they requires $O(n)$ group MSM for committing.

Hash-based schemes [33], [34], [35] usually relies on error-correcting code. These schemes uses error-correcting code like RS code to first encode the polynomial and construct a Merkle tree to commit it. Verifier can make sure the committed vector approximates a valid codeword through a constant number of samples. Their concrete prover time is much faster, as both encoding and hash are much cheaper than group operations. The cost is existing hash-based schemes has at least $O(\lambda \log^2 n)$, where λ is query number ranging from tens to thousands, depending on the specific choice of error-correcting code.

III. Preliminary

We use $\mathbb{F}$ to denote prime field with large multiplicative subgroup. For vector $\vec{v} \in \mathbb{F}^n$, we use $v[i]$ to denote its i-th element, starting from 0 by default. For matrix $V \in \mathbb{F}^{m \times n}$, we use $V[i, j]$ to denote the element on the i-th row, j-th column. For vectors $\vec{a}, \vec{b} \in \mathbb{F}^n$, we use $\vec{a} \odot \vec{b}$ to denote their Hadamard product, use $\vec{a} \otimes \vec{b}$ to denote their tensor product. We use $[n]$ to denote the set $\{0, \cdots, n-1\}$.

Merkle tree A Merkle tree consists of three algorithms.

- $mt \leftarrow \mathsf{MT.Construct}(\vec{v})$ outputs the whole Merkle tree mt, using $mt.\mathsf{Root}$ to denote the root of mt, as the commitment of vector $\vec{v}$.
- $(v_i, path_i) \leftarrow \mathsf{MT.Open}(i, mt)$ takes as input the query location i and merkle tree mt, outputs the i-th leaf v_i and its verification path $path_i$.

- $rt \leftarrow \mathsf{MT.Recover}(path_i, v_i, i)$ takes as input the merkle tree path, value v_i and query location, outputs the root of merkle tree. Verifier can check the equality of rt and previous merkle root commitment, to determine whether accept the given value.

Merkle tree is instantiated by collision-resistant and non-invertible hash functions.

A. Useful facts

Lemma 1 (Multilinear extension)**.** *A multilinear polynomial is a multivariate polynomial in which the degree of each variable is at most one. For every function $f : \{0,1\}^\mu \to \mathbb{F}$, there is a unique multilinear polynomial $\tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \cdots, X_\mu]$ such that $\tilde{f}(\vec{b}) = f(\vec{b})$ for all $\vec{b} \in \{0,1\}^\mu$. We call $\tilde{f}$ the multilinear extension of f, and $\tilde{f}$ can be expressed as*

$$\tilde{f}(\vec{X}) = \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b}) \cdot \tilde{eq}_{\vec{X}}(\vec{b})$$

where $\tilde{eq}_{\vec{X}}(\vec{b}) = \prod_{i \in [\mu]} \left(\vec{b}[i]\vec{X}[i] + (1 - \vec{b}[i])(1 - \vec{X}[i])\right)$.

Lemma 2 (Schwartz-Zippel Lemma)**.** *Let $f \in \mathbb{F}[X_1, \cdots, X_\mu]$ be a non-zero polynomial of total degree d over field $\mathbb{F}$. Let S be any finite subset of $\mathbb{F}$, and let $r_1, \cdots, r_\mu$ be μ field elements selected independently and uniformly from S. Then*

$$\Pr\left[f(r_1, \cdots, r_\mu) = 0\right] \leq \frac{d}{|S|}$$

Lemma 3 (Twin Polynomials)**.** *For μ-variate multilinear polynomial $\tilde{f}(X_1, \cdots, X_\mu)$, we denote univariate polynomial $f(X) \in \mathbb{F}^{<2^\mu}[X]$ which shares the same coefficients with $\tilde{f}$. More precisely, there exists a coefficient vector $\vec{f}$ of 2^μ length such that: for each $(x_1, \cdots, x_\mu) \in \mathbb{F}^\mu$,*

$$\tilde{f}(x_1, \cdots, x_\mu) = \langle \vec{f}, (1, x_1) \otimes \cdots \otimes (1, x_\mu) \rangle$$

and for each $x \in \mathbb{F}$, $f(x) = \langle \vec{f}, (1, x, x^2, \cdots, x^{2^\mu - 1}) \rangle$. We call $\tilde{f}$ and f are twin polynomials of each other. For $x \in \mathbb{F}$, there has $f(x) = \tilde{f}(x^{2^0}, x^{2^1}, \cdots, x^{2^{\mu-1}})$.

We naturally extend the definition of twin polynomials to bivariate polynomials: we call $f(X, Y) \in \mathbb{F}[X, Y]$ with X degree $2^\ell - 1$ and Y degree $2^\kappa - 1$ and $\mu := \ell + \kappa$-variate multilinear polynomial $\tilde{f}$ twin polynomials to each other if there exists a coefficient vector $\vec{f}$ of 2^μ length such that for all $x, y \in \mathbb{F}, (x_1, \ldots, x_\mu) \in \mathbb{F}^\mu$:

$$f(x, y) = \left\langle \vec{f}, (1, x, x^2, \ldots, x^{2^{\ell-1}}) \otimes (1, y, y^2, \ldots, y^{2^{\kappa-1}}) \right\rangle$$

$$\tilde{f}(x_1, \ldots, x_\mu) = \left\langle \vec{f}, (1, x_1) \otimes \ldots \otimes (1, x_\mu) \right\rangle$$

B. Erasure Code

Erasure Code. A function $E : \Gamma^k \to \Lambda^n$ is a (n, k, t)-linear erasure code if there is a deterministic algorithm $\mathsf{Reconst}$, such that for any $m \in \Gamma^k$, and any $I \subseteq [n]$ with $|I| \geq t$ we have $\mathsf{Reconst}((\hat{m}_i)_{i \in I}) = m$ for $\hat{m} := E(m)$.

Reed-Solomon Code. Reed-Solomon (RS) code [36] is one of the most widely used erasure codes. Roughly, it corresponds to evaluations of polynomials. More precisely, given an ordered set $E = (e_1, \cdots, e_n) \subseteq \mathbb{F}$, an RS code $\mathsf{RS}[k, n, \mathbb{F}] : \mathbb{F}^k \to \mathbb{F}^n$ works as follows. To encode a given message $\vec{m} \in \mathbb{F}^k$, interpret $\vec{m}$ as a degree $k - 1$ univariate polynomial f over $\mathbb{F}$. Next, f is evaluated at all points in E, leading to the codeword $\vec{c} = (f(e_1), \cdots, f(e_n))$. Reed-Solomon codes are maximum distance separable codes, which means any k symbols in $\vec{c}$ can reconstruct the original message using Lagrange interpolation.

Interleaved Reed-Solomon Codes. Given $\mathsf{RS}[k, n, \mathbb{F}]$, we construct a new code $\mathsf{RS}^{\equiv L}[k, n] : \mathbb{F}^{L \times k} \to (\mathbb{F}^L)^n$ as follows. To encode a message $M \in \mathbb{F}^{L \times k}$, write it as $M = (m^{(0)}, \cdots, m^{(L-1)})$, where each $m^{(i)} \in \mathbb{F}^k$ for $i \in [L]$. Then, for each $i \in [L]$, encode $m^{(i)}$ into $\hat{m}^{(i)} \in \mathbb{F}^n$ using the original RS code. Next, for each $j \in [n]$, the j-th symbol of the encoded message is given by $(\hat{m}^{(0)}[j], \cdots, \hat{m}^{(L-1)}[j]) \in \mathbb{F}^L$. It is straightforward to see that $\mathsf{RS}^{\equiv L}$ maintains the same recovery threshold as RS, since decoding can be achieved by independently invoking the reconstruction algorithm of RS on each of the L codewords.

C. Reed-Solomon Code Commitment

We recall the definition of erasure code commitments from [13]. As this work and existing data availability schemes only focus on Reed-Solomon code, we present RS code commitment here. In a nutshell, such a commitment scheme allows to commit to an interleaved RS codeword $c \in \mathsf{RS}^{\equiv L}$.

Definition 1. *Let $\mathsf{RS}^{\equiv L} : \mathbb{F}^{L \times k} \to (\mathbb{F}^L)^n$ be an interleaved Reed-Solomon code. An erasure code commitment scheme for $\mathsf{RS}^{\equiv L}$ is a tuple $\mathsf{CC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ of PPT algorithms, with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{gp}$ *takes as input the security parameter λ and outputs the global parameter* gp.
- $\mathsf{Commit}(\mathsf{gp}, m) \to (\mathsf{com}, St)$ *takes as input a global parameter* gp *and a message $m \in \mathbb{F}^{L \times k}$ and outputs a commitment* com *and a state St.*
- $\mathsf{Open}(\mathsf{gp}, St, i) \to \pi$ *takes as input a global parameter* gp*, a state St and an index $i \in [n]$, and outputs an opening π.*
- $\mathsf{Verify}(\mathsf{gp}, \mathsf{com}, i, \hat{m}_i, \pi) \to b$ *is deterministic, takes as input a global parameter* gp*, a commitment* com*, and index $i \in [n]$, a symbol $\hat{m}_i \in \mathbb{F}^L$, and an opening τ, and outputs a bit $b \in \{0, 1\}$.*

A RS code commitment scheme should satisfy completeness, position-binding and code binding. Their specific definitions are presented in Appendix A.

From code commitment to data availability. [13] proposed a general compiler that transforms any erasure code commitment into a secure DAS scheme. Since the concrete cost is largely dominated by the cost of CC, this motivates us to investigate this cryptographic primitive.

D. Polynomial Commitment

Definition 2 (Polynomial Commitment Scheme)**.** *A polynomial commitment scheme* PC *consists of a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$*:*

- $\mathsf{Setup}(1^\lambda, \mu) \to \mathsf{pp}$. *It takes the security parameter λ and $\mu \in \mathbb{N}$ (i.e., the number of variables in a polynomial) and outputs the public parameter* pp.

- $\mathsf{Commit}(\mathsf{pp}, \tilde{f}) \to (\mathcal{C}, \mathcal{D})$. *It takes a μ-variate multilinear polynomial $\tilde{f}$ and outputs a commitment $\mathcal{C}$ along with some auxiliary message $\mathcal{D}$ (e.g., some randomness).*
- $\mathsf{OpenPoly}(\mathsf{pp}, \mathcal{C}, \tilde{f}, \mathcal{D}) \to b$. *It takes the commitment $\mathcal{C}$, the multilinear polynomial f, and the auxiliary message $\mathcal{D}$ to verify the commitment and outputs a bit $b \in \{0, 1\}$.*
- $\mathsf{Eval}(\mathsf{pp}, \mathcal{D}, \vec{z}, \tilde{f}) \to (y, \pi)$. *It takes as input auxiliary message $\mathcal{D}$ generated in commitment, evaluation point $\vec{z}$ and polynomial $\tilde{f}$, outputs evaluation result y and evaluation proof π.*
- $\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, \vec{z}, y, \pi) \to 0/1$. *It takes as input commitment $\mathcal{C}$, evaluation point $\vec{z}$, result y and proof π, outputs a bit $b \in \{0, 1\}$.*

The PCS satisfies *completeness* if for any multilinear polynomial $\tilde{f} \in \mathbb{F}[X_1, \cdots, X_\mu]$ and any point $\vec{z} \in \mathbb{F}^\mu$, the following probability is 1:

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mu) \\ (\mathcal{C}, \mathcal{D}) \leftarrow \mathsf{Commit}(\mathsf{pp}, \tilde{f}) \end{array} : \mathsf{Eval}(\mathsf{pp}, \mathcal{C}, \vec{z}, \tilde{f}(\vec{z}); \tilde{f}, \mathcal{D}) = 1\right]$$

It is *binding* if for any $\mu \in \mathbb{N}$ and PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{r|r} & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mu) \\ b_0 = b_1 = 1 \wedge & (\mathcal{C}, \tilde{f}_0, \mathcal{D}_0, \tilde{f}_1, \mathcal{D}_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \tilde{f}_0 \neq \tilde{f}_1 & b_0 \leftarrow \mathsf{OpenPoly}(\mathsf{pp}, \mathcal{C}, \tilde{f}_0, \mathcal{D}_0) \\ & b_1 \leftarrow \mathsf{OpenPoly}(\mathsf{pp}, \mathcal{C}, \tilde{f}_1, \mathcal{D}_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

The scheme satisfies *(knowledge) soundness* if Eval is an argument (of knowledge) for the relation $\mathcal{R}_{\mathsf{Eval},\mathsf{pp}}$ defined as follows:

$$\left\{(x = \{\mathcal{C}, \vec{z}, y\}); w = \{\tilde{f}, \mathcal{D}\} : \begin{array}{r} \tilde{f}(\vec{z}) = y \\ \mathsf{OpenPoly}(\mathsf{pp}, \mathcal{C}, \tilde{f}, \mathcal{D}) = 1 \end{array}\right\}$$

Partial evaluation For a bivariate polynomial $f(X, Y)$, we define $v(X) = f(X, y)$ as the partial evaluation of a bivariate polynomial at $y \in \mathbb{F}$. One can prove the correctness of partial evaluation that $v(X)$ is the correct partial evaluation of the committed polynomial $f(X, Y)$ by proving the evaluation $f(\beta, y) = v(\beta)$ for a random $\beta \in_R \mathbb{F}$ sampled by the verifier.
Batched evaluation. Batched evaluation of multilinear PCS proves multiple evaluations in a single proof. Its syntax can be written as

$$\mathsf{BatchEval}(\mathsf{pp}, \tilde{f}, \left\{\vec{z}^{(i)}\right\}_{i \in [m]}) \to (\{y_i\}_{i \in [m]}, \pi)$$
$$\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, \left\{\vec{z}^{(i)}\right\}_{i \in [m]}, \{y_i\}_{i \in [m]}, \pi) \to 0/1$$

We introduce sumcheck protocol in Appendix B. Batched evaluation of multilinear PCS can be achieved by invoking the following sumcheck:

$$\sum_{i \in [m]} \left(\alpha^i \cdot \sum_{\vec{b} \in \{0,1\}^\mu} \left(\tilde{f}(\vec{b}) \cdot \tilde{eq}_{\vec{z}^{(i)}}(\vec{b})\right)\right)$$
$$= \sum_{\vec{b} \in \{0,1\}^\mu} \tilde{f}(\vec{b}) \cdot \left(\sum_{i \in [m]} \alpha^i \cdot \tilde{eq}_{\vec{z}^{(i)}}(\vec{b})\right) \stackrel{?}{=} \sum_{i \in [m]} \alpha^i \cdot y_i$$

where $\alpha \in \mathbb{F}$ is a verifier sampled random scalar. The prover time is $O(m \cdot n)$, and the proof size is only $O(\log n)$.
Multi-evaluation. Multi-evaluation is often confused with batched evaluation but is actually a distinct variant of $\mathsf{PC.Eval}$ in Definition 2. Specifically, in multi-evaluations, the prover generates n proofs for n evaluations, each of which can be verified independently:

$$\mathsf{MultiEval}(\mathsf{pp}, \tilde{f}, \left\{\vec{z}^{(i)}\right\}) \to \{(y_i, \pi_i)\}$$
$$\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, \vec{z}^{(i)}, y_i, \pi_i) \to 0/1$$

The most distinct difference between batched evaluation and multi-evaluation is that batched evaluation aims to optimize the proof size, whereas multi-evaluation focuses on reducing the prover's time.
Multi-partial evaluation. A hybrid variant is proving correct partial evaluations (instead of full evaluations) of the same polynomial on n points, producing n proofs. For simplicity, we present using bivariate polynomials $f(X, Y)$ and its partial evaluations $v_i(X) = f(X, y_i)$:

$$\mathsf{MultiPartialEval}(\mathsf{pp}, f, \{x_i\}) \to \{(v_i(X), \pi_i)\}$$
$$\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, x_i, v_i(X), \pi_i) \to 0/1$$

Connection to RS code commitment. Since the RS code is closely related to univariate polynomials, a multi evaluation of PCS naturally corresponds to a RS code commitment. More precisely, given an $\mathsf{RS}[k, n]$ code defined over a domain $\Omega \subset \mathbb{F}$ of size n, the prover $\mathcal{P}$ can commit to the encoding of a message $\vec{m} \in \mathbb{F}^k$ by committing to a polynomial $f(X)$ with degree less than k. To generate a proof for each symbol, the prover only needs to prove the *multi evaluations* of $f(x)$ for all $x \in \Omega$.

IV. Technical Overview

In this section, we briefly introduce our interleaved RS code commitment scheme. The core idea is a novel multi-partial evaluation framework composed of two main components: evaluation consolidation and a multilinear PCS. In Section IV-A, we present our evaluation consolidation scheme, which can be combined with any multilinear PCS to construct a CC scheme. In Section IV-B, we introduce $\mathsf{LightLigero}$, a new multilinear PCS specifically designed for information dispersal. Compared to existing RS code-based polynomial commitment schemes, our scheme achieves significantly smaller proof sizes.

A. Efficient Multi-partial Evaluation

As mentioned before, RS code commitment can be implemented with polynomial commitments: parse the data $\in \mathbb{F}^k$ into a univariate polynomial $f(X) \in \mathbb{F}^{(<k)}[X]$, commit it, and invoke the multi-evaluation protocol (see Section III-D) over the domain $\Omega = \left\{\omega^i\right\}_{i \in [n]}$, where ω is an n-th root of unity. Equivalently, we can interpolate the same data into a μ-variate multilinear polynomial $\tilde{f}$, the twin polynomial of f with $2^\mu = k$, then commit and run $\mathsf{MultiEval}$ on $\tilde{f}$ instead. From Lemma 3, evaluating $f(\omega^i)$ for the i-th replica

is equivalent of evaluating $\tilde{f}(\omega^i, \omega^{2i}, \ldots, \omega^{2^{\mu-1}\cdot i})$. Naively, the prover can run n instances of the evaluation protocol in parallel: each taking $O(n)$-time (with $k = \Theta(n)$), thus $O(n^2)$ prover time in total, which is impractical.

We propose a paradigm to achieve efficient multi-evaluation by first reducing n evaluations into a single random evaluation, and then invoking PC.Eval to prove the reduced evaluation. Effectively, through this paradigm a MultiEval task can be simplified into an Eval task: the prover only needs to generate PCS evaluation proof of a single point to convince n verifiers of their n evaluations.

It is noteworthy that prover needs to generate n reduction proof, each of which is supposed to guarantee the soundness of an one-to-one reduction. Previously, one-to-one reduction in Basefold [37] and HyperNOVA [38] is performed by proving the sumcheck $\sum_{\vec{b}} \tilde{f}(\vec{b}) \cdot \tilde{eq}_{\vec{z}}(\vec{b}) = \tilde{f}(\vec{z})$. However, sumcheck is difficult to amortize in the context of multi-evaluation, as it involves a different $\vec{z}$ for different sumcheck instances.

Our evaluation consolidation technique starts with a one-to-one *evaluation reduction* where the statement about the evaluation of a single fixed point is reduced to the evaluation of a random point on the same committed polynomial. This reduction scheme enables a consolidation process where the randomized evaluation points in multiple concurrent reductions are gradually restricted to the same point.

Evaluation reduction: fixed to random evaluation. Denote the statement $(x := \{\mathcal{C}, \vec{z}, y\}, w := \{\tilde{f}\}) \in \mathcal{R}_{\mathsf{Eval}}$ an instance of the PCS evaluation relation (defined in Section III-D). We show a μ-round interactive reduction to another statement $(x' := \{\mathcal{C}, \vec{r}, y_r\}, w' := w) \in \mathcal{R}_{\mathsf{Eval}}$ where $\vec{r}$ is a random point sampled by the verifier.

In the i-th ($i \in [1, \mu]$) round, $\mathcal{P}$ sends a linear function $l_i(X)$ to $\mathcal{V}$ who checks its consistency with the previous claim $l_i(z_i) = l_{i-1}(r_{i-1})$[4] and samples a new $r_i \in \mathbb{F}$ to $\mathcal{P}$, where

$$l_i(X) := \tilde{f}(r_1, \ldots, r_{i-1}, X, \vec{z}_{[i+1:]})$$

effectively reducing the claim from $\tilde{f}(r_1, \ldots, r_{i-1}, \vec{z}_{[i:]}) = l_{i-1}(r_{i-1})$ to $\tilde{f}(r_1, \ldots, r_i, \vec{z}_{[i+1:]}) = l_i(r_i)$. After μ rounds, we reduce the claim from $\tilde{f}(\vec{z}) = y$ to $\tilde{f}(\vec{r}) = y_r$ for a random vector $\vec{r}$.

The soundness of our reduction is reminiscent of the sumcheck protocol [39]. The high level idea is that if the claim of $l_{i-1}(r_{i-1})$ isn't correct, then $l_i(X)$ isn't correct as $l_i(z_i) = l_{i-1}(r_{i-1})$, and thus for a random $r_i \in \mathbb{F}$, there has $l_i(r_i)$ isn't correct with overwhelming probability. As a result, the original claim of $\tilde{f}(\vec{z})$ can be finally reduced to the claim of $\tilde{f}(\vec{r})$, replacing one variable in each round.

Moreover, our reduction is *PCS-agnostic*, as it solely reduces statements about polynomial evaluations without relying on specific evaluation protocols. This agnosticism enables flexible tradeoffs when instantiated with various multilinear polynomial commitment schemes, unlike previous approaches that are tightly coupled to IOPP-based [15] or KZG-based commitments [22], [29].

Evaluation consolidation: many fixed to single random. Ultimately, we aim to reduce many evaluation statements to one. Towards this goal, the first key observation is that: if the random r_i in each round is identical for all evaluation reductions, then the number of unique evaluation claims halves every round (see Figure 2).

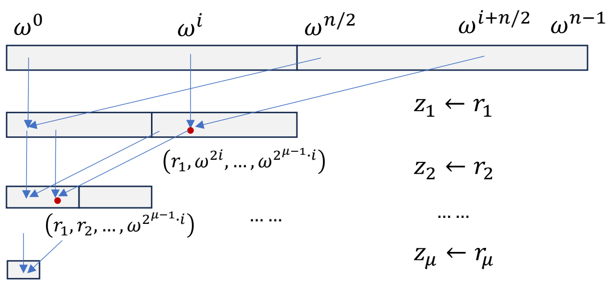

Fig. 2: Evaluation consolidation: the i-th evaluation is given by $f(\omega^i) = \tilde{f}(\omega^i, \omega^{2i}, \ldots, \omega^{2^{\mu-1}\cdot i})$. These evaluation points of $\tilde{f}$ are halved in each round of randomization until they are consolidated into a single random $\vec{r} \in \mathbb{F}^\mu$.

Specifically, when evaluating over the multiplicative group Ω, the evaluation point for the i-th replica $(\omega^i, \omega^{2i}, \ldots, \omega^{2^{\mu-1}\cdot i})$ only differs from that for the $(i+n/2)$-th replica in the first variable. Thus, after the first round of randomization where the first variable is replaced by a random challenge $X_1 \leftarrow r_1$, the two evaluation points fold into the same point: $(r_1, \omega^{2i}, \ldots, \omega^{2^{\mu-1}\cdot i})$. Similarly, in the m-th round, for all $j \in [2^m]$, messages sent to the $(i + \frac{j \cdot n}{2^m})$-th replica are identical in our evaluation reduction.

Secondly, to ensure a *shared* random verifier challenge r_i across n concurrent one-to-one reductions for $\{\vec{z_i}\}$, we use existing techniques [23] to modify the Fiat-Shamir transformation slightly. In every round, the prover builds a Merkle tree over n transcript-generated randomness and uses the root as the shared challenge for all verifiers. To convince $\mathcal{V}_i$ of the correct accumulation of its original random coin, $\mathcal{P}$ attaches the corresponding Merkle path when sending the merged challenge. Based on our observation that only $n/2^m$ different messages are sent in the m-th round, the leave number of Merkle tree used to derive the shared challenge halves every round, and thus the total number of hash operations performed by the prover is reduced to $O(n)$.

Reducing communication overhead. We propose an optimization to reduce round complexity, thereby reducing the communication cost by another $\log\log n$ factor. Specifically, to decrease the round complexity of evaluation reduction, $\mathcal{P}$ can send a s-variate multilinear polynomial of degree in each round instead of a linear function. Intuitively, this allows randomization of $s > 1$ variables per round, lowering the total number of rounds to $\frac{\mu}{s}$. More precisely, in the i-th round, the prover sends $\tilde{l}_i(X_1, \cdots, X_s)$ as:

$$l_i(X) = \tilde{f}(r_1, \ldots, r_{i-1}, X, \vec{z}_{[i+1:]}) \qquad \text{plain: } s = 1$$
$$\tilde{l}_i(X) = \tilde{f}(r_1, \ldots, r_{(i-1)\cdot s}, X_1, \cdots, X_s, \vec{z}_{[is+1:]}) \qquad \text{now: } s > 1$$

[4]The initial value set to the claimed multilinear evaluation $l_0(X) = \tilde{f}(\vec{z}) = y$; the final value is the evaluation at the randomized point $l_{\mu+1}(X) = l_\mu(r_\mu) = y_r$.

while the verifier samples a random challenges and checks consistency as: $\tilde{l}_i\left(\vec{z}_{[(i-1)\cdot s+1, i\cdot s]}\right) \overset{?}{=} \tilde{l}_{i-1}\left(\vec{r}_{[(i-1)\cdot s+1, i\cdot s]}\right)$. As a result, the soundness error of our reduction increases from $\frac{\mu}{|\mathbb{F}|}$ to $\frac{\mu}{s} \cdot \frac{2^s}{|\mathbb{F}|}$, but still remains negligible. To keep the size of $\tilde{l}_i(X)$ manageable, we set $s = O(\log\log n)$, resulting in a total proof size of $O(\log^2 n / \log\log n)$.

Interleaved RS code commitment. Generalizing from RS codes to interleaved RS codes $\mathsf{RS}^{\equiv L}$, one can naively run the evaluation consolidation on L polynomials concurrently. This naive scheme incurs L-multiplicative blow-up in communication costs. We propose to parse the entire message $M \in \mathbb{F}^{L\times k}$ as a *bivariate* polynomial $f(X, Y)$ where $\deg_X(f) < k, \deg_Y(f) < L$. The codeword contains partial evaluations over the multiplicative group Ω.

In interleaved RS code commitment, each replica receives the i-th symbol $v_i(X) \in \mathbb{F}^{<L}[X]$, representing the partial evaluation $f(X, \omega^i)$. As a result, the prover needs to run a MultiPartialEval over Ω on $f(X, Y)$. As explained in Section III-D, we can prove the correctness of $v_i(X)$ via proving $v_i(\beta) = f(\beta, \omega^i)$ for a random $\beta \in \mathbb{F}$. Generating multiple evaluation proofs for $\{v_i(\beta)\}_{i\in[n]}$ is just another MultiEval on the univariate $f(\beta, Y)$ whose twin polynomial can benefit from the preceding evaluation consolidation technique.

Putting it all together, we manage to consolidate n PartialEval statements, corresponding to the L-sized symbols opened to every replica in interleaved RS code, into a single statement $\tilde{f}(\vec{\beta}, \vec{r}) = y_r$ with $\vec{\beta} = (\beta, \beta^2, \ldots, \beta^{L/2})$. The consolidated statement can be proven using a single evaluation proof from the chosen multilinear polynomial commitment.

B. RS code-based PCS with $O(\lambda \log n)$ proof size

Multilinear PCS is a key building block in our code commitment scheme, and we observe that existing PCS will all impose a significant overhead on either computational or communication costs. To address this issue, we provide an overview of a PCS specifically tailored for the desirable cost profile in DA applications.

Existing polynomial commitment schemes can be roughly categorized into two classes: group-based and code-based. Generally, code-based PCS achieves significantly faster prover time, as encoding and constructing Merkle trees is much cheaper than performing multi-scalar multiplication (MSM) in groups. However, this advantage comes at the cost of larger proof sizes. Specifically, code-based PCS requires committing to $O(\log n)$ Merkle trees, each of which needs to open λ paths, resulting in an overall proof size of $O(\lambda \log^2 n)$.

To illustrate this trade-off, consider two representative schemes from each category: PST [30] (group-based) and Deepfold [35] (code-based). For a polynomial of length 2^{20}, PST achieves a prover time of approximately 8 seconds with a proof size of about 1KB. In contrast, Deepfold achieves a prover time of roughly 2 seconds, but with a significantly larger proof size of around 300KB (assuming a code rate of 1/4). Here, we present a multilinear PCS scheme which can combine their advantage, with roughly 2.5s prover time and 50KB proof size.

Our solution is inspired by Ligero++ [34]. To reduce the evaluation cost in Virgo [40], a RS code-based multilinear PCS, Ligero++ proposes encoding the polynomial into an interleaved RS codeword instead of using a single-layer RS code. Specifically, the prover arranges the polynomial coefficients into a $\log n \times \frac{n}{\log n}$ matrix C, and encodes it into an interleaved RS codeword $\mathsf{RS}^{\equiv \log n}$. The resulting codeword is committed using a Merkle tree, where each column of C corresponds to a leaf node.

Let $\mu' = \log\log n$. To prove the evaluation of $\tilde{f}(z_1, \cdots, z_\mu)$, it suffices to prove the evaluation of $\tilde{f}'(z_{\mu'+1}, \cdots, z_\mu)$, where the coefficients of $\tilde{f}'$ are computed by linearly combining the rows of C using z_1 through $z_{\mu'}$. Recall that Virgo is a code-based PCS scheme that performs evaluation by opening selected entries of a single-layer RS codeword at random positions. With the Merkle tree commitment of the interleaved RS codeword, it becomes straightforward to retrieve the codeword of f'. Specifically, to open $\tilde{f}'$, suppose Virgo requires querying entries $\vec{c}[i]$ of the codeword $\vec{c}$, which encodes the coefficients of $\tilde{f}'$. This can be achieved by querying the i-th column of C and performing the necessary linear combination on the verifier side. This approach effectively reduces the evaluation cost of Virgo by a factor of $\log n$, while maintaining the same proof size of $O(\lambda \log^2 n)$ as in the original Virgo scheme.

However, transplanting this idea to a setting that combines RS codes with PST is challenging. The primary challenge lies in the fact that PST is not a code-based PCS, which makes it difficult to integrate as seamlessly as RS codes are integrated with Virgo. A natural idea to address this issue is to invoke PST.Commit to commit to $\tilde{f}'$, and then prove the evaluation $\tilde{f}'(z_{\mu'+1}, \cdots, z_\mu)$. However, this approach is clearly insufficient, as it does not verify the consistency between $\tilde{f}'$ and the original Merkle tree commitment of C.

The above consistency check problem can be summarized as follows: given a vector committed via a Merkle tree, determine whether it approximates the codeword corresponding to a polynomial committed by PST. We show that this consistency can be verified by sampling λ random columns from the codeword and performing batch evaluations of $\tilde{f}'$ at these λ points. The total prover time, including both commitment and evaluation, is $O\left(n \log n \cdot \mathbb{F} + n \cdot \mathbb{H} + \frac{n}{\log} \cdot \mathbb{G}\right)$, where $n \log n \cdot \mathbb{F} + n \cdot \mathbb{H}$ is from encoding and committing C, $\frac{n}{\log} \cdot \mathbb{G}$ is from committing and opening PST. The proof size is $O(\lambda \log n)$, as it involves opening λ columns in the matrix C.

V. Interleaved RS Code Commitment

In this section, we present our interleaved RS code commitment scheme, i.e. multi-partial evaluation for any multilinear PCS. This multi-evaluation leverages a key tool called the *evaluation reduction* protocol.

A. Evaluation Reduction

Given the commitment of a multilinear polynomial $\tilde{f}(X_1, \ldots, X_\mu)$, $\mathcal{V}$ can reduce a purported claim $\tilde{f}(\vec{z}) = y$

Protocol 1. Multilinear Evaluation Reduction.
Public input: a point $\vec{z} = (z_1, \cdots, z_\mu)$, and its purported evaluation $y \in \mathbb{F}$ (of $\tilde{f}$ on $\vec{z}$).
Prover witness: the multilinear polynomial $\tilde{f}$.
Verifier Output: a random point $\vec{r} \in \mathbb{F}^\mu$ and its purported evaluation y_r of the same polynomial $\tilde{f}$ on $\vec{r}$.

1) $\mathcal{P}$ sends a linear function $l_1(X) = \tilde{f}(X, \vec{z}_{[2:]})$ to $\mathcal{V}$.
2) $\mathcal{V}$ checks $l_1(z_1) = y$, outputs $\perp$ if failed.
3) The i-th round, $i \in \{2, \cdots, \mu\}$:
   a) $\mathcal{V}$ randomly samples $r_{i-1} \in \mathbb{F}$ and sends it to $\mathcal{P}$.
   b) $\mathcal{P}$ sends a linear function to $\mathcal{V}$:
   $$l_i(X) = \tilde{f}(\vec{r}_{[:i-1]}, X, \vec{z}_{[i+1:]}).$$
   c) $\mathcal{V}$ checks $l_{i-1}(r_{i-1}) = l_i(z_i)$, outputs $\perp$ if failed.
4) $\mathcal{V}$ samples r_μ and outputs $\vec{r} = (r_1, \ldots, r_\mu)$ and $y_r = l_\mu(r_\mu)$.

to another claim of the evaluation of $\tilde{f}(\vec{r}) = y_r$ with the help of $\mathcal{P}$, where $\vec{r} \in \mathbb{F}^\mu$ is a random point generated by $\mathcal{V}$ during interaction. We call this procedure *evaluation reduction* with formal definition in Appendix C.

Evaluation reduction can be achieved by employing sumcheck to prove $\sum_{\vec{b} \in \{0,1\}^\mu} \tilde{f}(\vec{b}) \cdot \tilde{eq}_{\vec{z}}(\vec{b}) = y$ by leveraging Lemma 1. In the last step, $\mathcal{V}$ needs oracle access to $\tilde{f}(\vec{r}) \cdot \tilde{eq}_{\vec{z}}(\vec{r})$ for a random $\vec{r}$. Since $\tilde{eq}_{\vec{z}}(\vec{r})$ can be computed locally, $\mathcal{P}$'s task has been reduced from proving the value of $\tilde{f}(\vec{z})$ to proving the value of $\tilde{f}(\vec{r})$ at the end of the sumcheck protocol. The above evaluation reduction protocol has been used in schemes like Basefold [37] and HyperNOVA [38].

We propose a strictly more efficient evaluation reduction in Protocol 1. Instead of sending a degree-2 univariate polynomial $g_i(X) = \sum_{\vec{b}_{[i+1:]}} f(\vec{r}_{[:i-1]}, X, \vec{b}_{[i+1:]}) \cdot \tilde{eq}_{\vec{z}}(\vec{r}_{[:i-1]}, X, \vec{b}_{[i+1:]})$ in i-th round, $\mathcal{P}$ in our simplified protocol only sends a linear function $l_i(X) = \tilde{f}(\vec{r}_{[:i-1]}, X, \vec{z}_{[i+1:]})$. The benefits of our protocol are two folds. Firstly, the reduction in degree of per-step polynomial reduces concrete prover cost and communication cost. Secondly, by eliminating the $eq_{\vec{z}}(\cdot)$ term, the prover message $l_i(X)$ in round i becomes independent of $\vec{z}_{[:i-1]}$, a property that enables evaluation consolidation where we extend this reduction to many $\{\vec{z}_j\}$. The security proof of this scheme is deferred to the next section.

B. Non-interactive Evaluation Consolidation

A non-interactive *evaluation consolidation* enables one prover to reduce n purported evaluations in parallel to a single (random) evaluation, generating n independent proofs. The i-th proof serves as an argument for the evaluation reduction corresponding to the i-th claim.

Although a single evaluation reduction in Protocol 1 can be rendered non-interactive via the Fiat-Shamir transform, ensuring a shared random point across all n reductions while preserving the soundness for each proof is non-trivial. Fortunately, [23] proposed a solution that uses a Merkle tree to accumulate all n messages in each round, treating the root as the shared challenge for all n reductions. The corresponding Merkle paths are included in the evaluation reduction proof provided to each verifier to ensure soundness.

Communication Optimization A limitation of the above scheme is its $O(\log^2 n)$ proof size, which arises from the round complexity of $O(\log n)$, with each round requiring a $O(\log n)$-sized Merkle path to derive the next challenge. The cumulative proof size across rounds motivates us to reduce the round complexity in our evaluation reduction protocol.

Our key observation is that the prover can send a higher degree polynomial instead of a linear function to $\mathcal{V}$ in each round. For example, $\mathcal{P}$ can send a univariate polynomial[5] $p(X) = \tilde{f}(X, X^2, X^4, \vec{z}_{[4:]})$ to $\mathcal{V}$ in the first round, and let $\mathcal{V}$ samples $r \in \mathbb{F}$ and reduce the original claim to $\tilde{f}(r, r^2, r^4, \vec{z}_{[4:]})$.

Assuming the univariate polynomial in each round is of degree $2^s - 1$, the evaluation consolidation only requires $\frac{\log n}{s}$ rounds. By setting $s = \log\log n$, this approach achieves a communication complexity of $O(\log^2 n / \log\log n)$. The optimized non-interactive evaluation consolidation procedure is presented in Protocol 2.

The following theorems establish the security properties of our optimized evaluation reduction scheme.

Theorem 1 (Completeness.). *If $\tilde{f}(\vec{z}) = y$, and both prover $\mathcal{P}$ and verifier $\mathcal{V}$ are honest, then $\tilde{f}(\vec{r}) = y_r$.*

Theorem 2 (Soundness.). *If $\tilde{f}(\vec{z}) \neq y$, the probability of $y_r = \tilde{f}(\vec{r})$ is* negl. *More precisely, when $\tilde{f}$ is a μ-variate multilinear polynomial, we have*

$$\Pr[\tilde{f}(\vec{r}) = y_r] \leq \frac{\mu \cdot 2^s}{s \cdot |\mathbb{F}|}$$

The completeness of evaluation reduction is straightforward. The soundness proof is deferred to Appendix E.

C. Interleaved RS Code Commitment

An RS code commitment scheme can be constructed by combining evaluation consolidation with any multilinear polynomial commitment scheme. By invoking the $\mathsf{mlPC.Eval}()$ only once at the consolidated random point, the overall prover cost is effectively amortized across all proofs. A notable advantage of our scheme over FRIDA [15] lies in its flexibility in selecting the multilinear PCS. As a PCS-agnostic scheme, it allows us to effectively balance prover time and proof size, rather than being constrained by the inherent characteristics of the IOPP used in FRIDA.

Our scheme can be straightforwardly extended to *interleaved* RS code commitment. For simplicity, we assume that parameters L, k, n in interleaved RS code $\mathsf{RS}^{\equiv L}[k, n]$ are all powers of 2. We observe that the original $L \times k$ matrix can be interpreted as a bivariate polynomial $f(X, Y)$, with

[5]In Section IV-A, we use multilinear polynomial, which is equivalent to high-degree polynomial due to lemma 3.

Protocol 2. Non-interactive Evaluation Consolidation NIEC

Notation: $s \in \mathbb{N}^+$ is a tunable parameter that determines the degree of univariate polynomial sent in each round of the evaluation reduction; $\omega \in \mathbb{F}$ is the n-the primitive roots of unity of the domain $\Omega = (\omega^0, \omega^1, \ldots, \omega^{n-1})$; $\mathsf{FS.add}(\cdot)$ appends to the Fiat-Shamir transcript, $\mathsf{FS.chal}()$ generates the next transcript-bounded verifier challenge.

- **Consolidate**$(\vec{e}, \Omega) \rightarrow (\vec{r}, \{\vec{\pi}_j\}_{j\in[n]})$: Given the evaluations $\vec{e}$ of a μ-variate multilinear polynomial $\tilde{f}$ over $\left\{\vec{z}_j = (\omega^j, \omega^{2j}, \ldots, \omega^{2^{\mu-1}\cdot j})\right\}_{j\in[n]}$ and the evaluation domain Ω, it outputs the consolidated evaluation point $\vec{r} \in \mathbb{F}^\mu$, the consolidation proofs $\vec{\pi}_j$ for every j-th replica.
  1) Initialize the transcript with public parameters $\mathsf{FS.add}(\Omega, s, n)$, and empty vectors $\{\vec{\pi}_j\}_{j\in[n]}$.
  2) For $j \in [n/2^s]$, $\mathcal{P}$ interpolates $p_{1,j}(X) = \tilde{f}(X, X^2, \ldots, X^{2^s}, \vec{z}_{j[s+1:]})$ of degree $(2^s - 1)$ from valuations $(\vec{e}[j + \frac{n\ell}{2^s}])_{\ell\in[2^s]}$ on points $(\omega^{j+\frac{n\ell}{2^s}})_\ell$; merklize them $mt \leftarrow \mathsf{MT.Construct}(\{p_{1,j}(X)\}_j)$, append to the transcript $\mathsf{FS.add}(mt.\mathsf{Root})$.
     $\mathcal{P}$ then append the merkle path $\mathsf{MT.Open}(j \mod \frac{n}{2^s}, mt)$ to each consolidation proof $\vec{\pi}_j$ for $\forall j \in [n]$.
  3) The i-th round, $i \in \{2, \ldots, \lfloor\frac{\mu}{s}\rfloor\}$:
     a) Derive the next challenge scalar $r_{i-1} \leftarrow \mathsf{FS.chal}()$
     b) For $j \in [n/2^{i\cdot s}]$, interpolates $p_{i,j}(X)$ from evaluations $(p_{i-1,j+\frac{n\ell}{2^{s\cdot i}}}(r_{i-1}))_{\ell\in[2^s]}$ on points $(\omega^{j+\frac{n\ell}{2^{s\cdot i}}})_{\ell\in[2^s]}$; merklize them $mt \leftarrow \mathsf{MT.Construct}(\{p_{i,j}(X)\}_j)$, append to the transcript $\mathsf{FS.add}(mt.\mathsf{Root})$.
     c) For $\forall j \in [n]$, append the merkle path $\mathsf{MT.Open}(j \mod \frac{n}{2^{s\cdot i}}, mt)$ to each consolidation proof $\vec{\pi}_j$.
  4) Derive the last challenge $r_{\lfloor\frac{\mu}{s}\rfloor} \leftarrow \mathsf{FS.chal}()$ and outputs $\vec{r} := (r_1, \ldots, r_{\lfloor\frac{\mu}{s}\rfloor})$ and all $\{\vec{\pi}_j\}_{j\in[n]}$.
- **Verify**$(\vec{z}_j, \vec{e}[j], \vec{r}, \vec{\pi}_j) \rightarrow b$: Given the evaluation point $\vec{z}_j$, its claimed evaluation as the j-th element in $\vec{e}$, and consolidated random point $\vec{r}$, and the proof $\vec{\pi}_j$ for all μ/s rounds, outputs 1 to accept, 0 to reject.
  1) Initialize the transcript with public parameters $\mathsf{FS.add}(\Omega, s, n)$
  2) Parse $(p_{1,j}(X), path_1) \leftarrow \vec{\pi}_j[0]$, check $p_{1,j}(\vec{z}_j[0]) \overset{?}{=} \vec{e}[j]$, outputs 0 if failed. Append the merkle root to transcript $\mathsf{FS.add}(\mathsf{MT.Recover}(path_1, p_{1,j}(X), j \mod \frac{n}{2^s}))$.
  3) The i-th round, $i \in \{2, \ldots, \lfloor\frac{\mu}{s}\rfloor\}$:
     a) Derive the next challenge $r_{i-1} \leftarrow \mathsf{FS.chal}()$
     b) Parse the next $(p_{i,j}(X), path_i) \leftarrow \vec{\pi}_j[i]$, check $p_{i-1,j}(r_{i-1}) \overset{?}{=} p_{i,j}(\vec{z}_j[s\cdot i])$, outputs 0 if failed.
     c) Update the transcript $\mathsf{FS.add}(\mathsf{MT.Recover}(path_i, p_{i,j}(X), j \mod \frac{n}{2^{s\cdot i}}))$.
  4) Derive the last challenge $r_{\lfloor\frac{\mu}{s}\rfloor} \leftarrow \mathsf{FS.chal}()$ and checks $\vec{r} \overset{?}{=} (r_1, \ldots, r_{\lfloor\frac{\mu}{s}\rfloor})$, outputs 0 if failed, otherwise outputs 1.

$\deg(X) = L - 1, \deg(Y) = k - 1$. The j-th encoded symbol can be obtained by the partial evaluation $v_j(X) = f(X, \omega^j)$. Let $\ell = \log L$, $\kappa := \log k$. There is a multilinear polynomial $\tilde{f}$ whose variable number is $\ell + \kappa$, such that $f(x, y) = \tilde{f}\left(x^{2^0}, \cdots x^{2^{\ell-1}}, y^{2^0}, \cdots, y^{2^{\kappa-1}}\right)$ for each $x, y \in \mathbb{F}$. Prover can commit this $\tilde{f}$ and leverage our evaluation consolidation technique to generate n partial evaluation proofs with low amortized cost.

More specifically, the prover uses Fiat-Shamir to derive a random $\beta \in \mathbb{F}$, and prove the evaluation of $\tilde{f}(\vec{\beta}, \vec{\omega}_j)$ for party j, where $\vec{\beta} = (\beta, \beta^2, \ldots, \beta^{2^{\ell-1}})$, $\vec{\omega}_j = (\omega^{2^0\cdot j}, \cdots, \omega^{2^{\mu-1}\cdot j})$. Subsequently, the prover can consolidate all n claims into a shared claim $\tilde{f}(\vec{\beta}, \vec{r})$, and invoke the multilinear PCS evaluation on $\vec{r}$. The full protocol is presented in Protocol 5, detailed in Appendix D. Soundness of Protocol 5 is implied by soundness of PCS and evaluation reduction, which can be summarized as the following corollary.

Corollary 1. *Protocol 5 is a multi-partial evaluation of multilinear PCS with negligible soundness error.*

Complexity.

- Prover time: Apart from the $O(L \cdot n \log n)$ field operations for encoding, $\mathcal{P}$ must commit to a multilinear polynomial of size $k \cdot L$, perform a single opening, and compute $O(n \cdot L)$ hashes during evaluation consolidation. Concretely, the evaluation consolidation cost $O(nL \cdot \mathbb{H})$ is much faster than both encoding and any existing PCS. When using Dory, the overall prover complexity is $O(n \cdot L) \cdot \mathbb{G}$, where group operations dominate due to their significantly higher cost compared to hashing.
- Communication: The communication overhead per-replica includes the multilinear PCS commitment, the consolidated evaluation point and its evaluation, the $O(\frac{\log^2 n}{\log\log n})$-sized evaluation consolidation proof, one PCS evaluation proof, and one $O(\log n)$-sized Merkle proof. The size of the PCS evaluation proof depends on the specific PCS chosen; for example, when using Dory [31], the PCS proof size is $O(\log(n \cdot L))$. Most importantly, the overall proof size is sublinear to both L and n.
- Verifier time: The verifier's complexity is equivalent to that of communication. In practice, the verifier's runtime is less than 0.1 seconds, so its actual performance is negligible.

Verifiable Secret Sharing. Our evaluation consolidation scheme can also benefit verifiable secret sharing (VSS). A significant difference in VSS is that the proof sent by $\mathcal{P}$ must ensure zero-knowledge for all information beyond the symbol itself. We detail how to achieve zero-knowledge and construct VSS scheme in Appendix F.

VI. LightLigero: RS CODE-BASED PCS WITH $O(\lambda \log n)$ PROOF SIZE

This section presents our polynomial commitment scheme that achieves $O(n \log n)\mathbb{F} + O(n)\mathbb{H} + O(\frac{n}{\log n})\mathbb{G}$ prover time and $O(\lambda \log n)$ proof size. The overall prover time is slightly larger than Deepfold but has much smaller proof size.

A. LightLigero *construction*

In LightLigero, the prover first decomposes the μ-variate multilinear polynomial $\tilde{f}(X_1, \cdots, X_\mu)$ into $\log n$ components, such that

$$\tilde{f}(X_1, \cdots, X_\mu) = \sum_{\vec{b} \in \{0,1\}^{\mu'}} \tilde{f}_{\vec{b}}(X_{\mu'+1}, \cdots, X_\mu) \cdot \prod_{i \in [\mu']} X_{i+1}^{\vec{b}[i]},$$

where each $\tilde{f}_{\vec{b}}(\cdot)$ is a multilinear polynomial, and there are $2^{\mu'} = \log n$ such polynomials, each of size $\frac{n}{\log n}$.

To commit to $\tilde{f}$, the prover encodes the coefficient vector of each $\tilde{f}_{\vec{b}}$ into a Reed-Solomon (RS) codeword, resulting in a $2^{\mu'} \times (2^{\mu - \mu'}/\rho)$ matrix, where ρ is the RS code rate. The prover then hashes the columns of this matrix into a Merkle tree and publishes the Merkle root as the commitment to the entire polynomial.

In order to convince $\mathcal{V}$ the evaluation of $\tilde{f}(z_1, \cdots, z_\mu) = y$, $\mathcal{P}$ commits $\mu - \mu'$ variate multilinear polynomial $\tilde{f}'$ by PST, where

$$\tilde{f}'(X_{\mu'+1}, \cdots, X_\mu) = \sum_{\vec{b} \in \{0,1\}^{\mu'}} \tilde{f}_{\vec{b}}(X_{\mu'+1}, \cdots, X_\mu) \cdot \prod_{i \in [\mu']} z_{i+1}^{\vec{b}[i]}$$

and proves $\tilde{f}'(z_{\mu'+1}, \cdots, z_\mu) = y$. Moreover, prover needs to prove the consistency of matrix C and $\tilde{f}'$, which requires opening λ leaves in Merkle tree and proving λ evaluations in $\tilde{f}'$. The formal protocol of interactive version of of multilinear PCS is presented in Protocol 3. This scheme can be transformed into *non-interactive* through Fiat-Shamir transformation.

B. Security Analysis

Theorem 3 (Soundness). *Assume the the subset $I \subseteq$ has size s, the soundness error of* LightLigero *is* $\rho^s + \mathsf{negl}(\lambda)$.

The proof is deferred to Appendix E. From Theorem 3, we can conclude that to achieve 100-bit security, prover can set $s = 50$ when $\rho = 1/4$.

Cost of LightLigero**.** Next, we analyze asymptotic and concrete cost of LightLigero.

- Prover time: The commitment cost of prover is $O(n \log n \cdot \mathbb{F} + n \cdot \mathbb{H})$, representing the cost of RS code encoding and constructing Merkle tree. Evaluation cost includes several parts. Committing and opening $\tilde{f}'$ requires $O(\frac{n}{\log n})\mathbb{G}$. Batched evaluating λ values over $\tilde{f}'$ cost includes $\frac{\lambda n}{\log n}$, which is both asymptotically and concretely much smaller than $O(n \log n \mathbb{F})$. Thus, the overall prover complexity is $O(n \log n \cdot \mathbb{F} + n \cdot \mathbb{H} + \frac{n}{\log n} \cdot \mathbb{G})$.
  Concretely, the prover time of LightLigero is comparable to state-of-the-art Reed-Solomon code-based PCS schemes. While PST is approximately $5\times$ slower than Deepfold, for $\log n = 16$, the overhead introduced by PST increases the total prover cost by only 30%.
- Proof size: Proof size of LightLigero comprises of PST proof size, batched evaluation proof and Merkle tree leaves and paths. PST proofs and batched evaluation proof are only $O(\log n)$. Merkle tree leaves and paths are all $O(\lambda \log n)$. The overall proof size is $O(\lambda \log n)$.
  Concretely, when $n = 2^{20}$, sample number $s = 50$, we set leaf size to be 16. Communication cost of all leaves is $50 \times 16 = 800$ scalars. Each Merkle path length is also 16, so the overall Merkle path is at most $50 \times 16 = 800$ hashes. Each hash and scalar are 32B, so the overall proof size is roughly $32\text{B} \times 1600 = 50\text{KB}$, at least $4\times$ smaller than Deepfold.
- Verifier time: Similar with proof size, the verifier complexity is $O(\lambda \log n)$. The concrete verifier time is at most 50ms, completely negligible compared to prover time.

VII. EVALUATION

This section presents experimental evaluations of our scheme. Our code has been open-sourced at https://0x0.st/8VOF.zip.

Hardware. All our experiments were run locally on MacBook M1 laptop with 16GM RAM using a single thread. No networking is required as we only benchmark the code commitment primitive.

A. Data availabilty

We have implemented all relevant code commitment schemes surveyed in Section II-A in Rust and conducted comprehensive experiments to demonstrate our improvements.

We first evaluate two concrete instantiations of our code commitment scheme CONDA against the main baseline FRIDA [15] (Crypto'24) in DA settings. Since industrial DA solutions are already handling 15MB blocks today[6], block sizes in our experiments range from 32MB to 128MB to accommodate near-future workloads. Given our goal of targeting thousands of nodes for decentralization[7], total number of nodes range from 1024 to 4096. Since the performance of code commitment schemes dominates that of the end-to-end DA protocols, we only benchmark efficiency metrics of the underlying code commitments. Moreover, we omit benchmarks on the reconstruction algorithm, because it is nearly identical in all DA protocols: using Lagrange interpolation to recover the original polynomial that embeds the block data.

[6]Celestia DA at 8MB blocks: https://blog.celestia.org/ginger/; EigenDA achieving 15MB/sec throughput: https://docs.eigenda.xyz/core-concepts/overview.

[7]Ethereum today has 6800 detected participating nodes as of April, 2025: https://etherscan.io/nodetracker.

Protocol 3. Multilinear polynomial commitment LightLigero

- $\mathsf{Commit}(\tilde{f}) \to \mathcal{C}$: Given the μ-variate multilinear polynomial $\tilde{f}$, prover generates commitment of $\tilde{f}$.
  1) Assume the coefficient of $\tilde{f}$ can be arranged into matrix $C \in \mathbb{F}^{2^{\mu'} \times 2^{\mu-\mu'}}$.
  2) Compute matrix $C' \in \mathbb{F}^{2^{\mu'} \times m}$ such that $i \in [2^{\mu'}]$, $C'[i,:] = \mathsf{RS}(C[i,:], \rho)$, where $m = 2^{\mu-\mu'}/\rho$.
  3) Construct Merkle tree mt with m leaves, the j-th leaf is μ'-variate multilinear polynomial $\tilde{p}_i$ whose coefficient is $C'[:,j]$. Output $\mathcal{C} = mt.\mathsf{Root}$ as commitment of $\tilde{f}$.
- $\left\langle \mathsf{Eval}(\tilde{f}), \mathsf{Verify}(\mathcal{C}) \right\rangle (\vec{z} = (z_1, \cdots, z_\mu), y) \to 0/1$: $\mathcal{P}$ takes the polynomial $\tilde{f}$ and the random evaluation point $\vec{z}$ as input, $\mathcal{V}$ takes commitment $\mathcal{C}$ as input, outputs 0/1 indicating whether accept the proof.
  1) $\mathcal{P}$ invokes $\mathcal{C}' \leftarrow \mathsf{PST.Commit}$ to commit $\mu - \mu'$ variate multilinear polynomial $\tilde{f}' = \tilde{f}(z_1, \cdots, z_{\mu'}, X_{\mu'+1}, \cdots, X_\mu)$. $\mathcal{P}$ sends $\mathcal{C}'$ to $\mathcal{V}$.
  2) Let $m = 2^{\mu-\mu'}/\rho$. $\mathcal{V}$ samples a random subset $I \subseteq [m]$ such that $|I| = s$, and sends I to $\mathcal{P}$. $\mathcal{V}$ samples uniform random $\alpha \in \mathbb{F}$ and sends α to $\mathcal{P}$.
  3) $\mathcal{P}$ opens $\{(\tilde{p}_i, path_i)\}_{i \in I} \leftarrow \mathsf{MT.Open}(I, mt)$ and sends them to $\mathcal{V}$. $\mathcal{V}$ checks the correctness of these Merkle paths. If one of these paths is incorrect, $\mathcal{V}$ outputs 0 and the protocol terminates.
  4) $\mathcal{P}$ and $\mathcal{V}$ perform batched open of $\tilde{f}'(z_{\mu'+1}, \cdots, z_\mu)$ and $f'(\omega^i)$ for all $i \in I$. Specifically, $\mathcal{P}$ and $\mathcal{V}$ perform sumcheck for
  $$\sum_{\vec{b} \in \{0,1\}^{\mu-\mu'}} \tilde{f}'(\vec{b}) \cdot \left(\tilde{eq}_{\vec{z}'}(\vec{b}) + \alpha \cdot \sum_{j \in [s]} \alpha^i \tilde{eq}_j(\vec{b}) \right) = y + \alpha \cdot \sum_{j \in [s]} \alpha^j \cdot y_i$$
  where $\vec{z}' := (z_{\mu'+1}, \cdots, z_\mu)$, $\tilde{eq}_j(\cdot) := \tilde{eq}_{\vec{\omega}^{(j)}}(\cdot)$, $\omega^{(j)} := (\omega^{I[j]}, \omega^{2I[j]}, \omega^{4I[j]}, \cdots, \omega^{2^{\mu-\mu'} \cdot I[j]})$. This sumcheck will reduce the batched evaluations to claimed evaluation for $\tilde{f}'(r_1, \cdots, r_{\mu-\mu'})$.
  5) $\mathcal{P}$ invokes $\mathsf{PST.Eval}$ to prove the evaluation of $\tilde{f}'(r_1, \cdots, r_{\mu-\mu'})$ and sends the proof to $\mathcal{V}$.
  6) If the proof can't pass, $\mathcal{V}$ outputs 0; otherwise outputs 1.

	$N = 1024$			$N = 2048$			$N = 4096$		
Block size $\lvert M \rvert$	32MB	64MB	128MB	32MB	64MB	128MB	32MB	64MB	128MB
Prover (s)									
FRIDA [15]	2.39	5.04	12.06	2.34	4.92	11.04	2.37	5.08	12.47
CONDA + Dory	8.80	16.87	34.50	8.98	17.16	33.15	8.94	17.24	34.09
CONDA + LightLigero	2.88	5.81	12.03	2.91	5.95	11.81	2.86	6.03	13.17
Per-node Comm. (KB)									
FRIDA	798.74	992.02	1316.70	734.74	864.02	1060.70	702.74	800.02	932.70
CONDA + Dory	131.98	260.08	516.17	67.98	132.08	260.17	36.63	68.73	132.82
CONDA + LightLigero	194.63	324.77	582.91	130.63	196.77	326.91	99.28	133.42	199.56
Verifier (ms)									
FRIDA	251	492	1091	240	495	1030	250	513	1032
CONDA + Dory	9	8	10	7	8	10	7	9	8
CONDA + LightLigero	13	21	36	14	22	36	12	21	34

TABLE IV: Comparison against the baseline FRIDA [15] (Crypto'24) across varying block sizes and network sizes. Single-threaded prover time includes the interleaved encoding, the codeword commit and proof generation for all codeword symbols. Per-node communication includes the fragment of symbols distributed to the node, the commitment, and the opening proofs for symbols within the fragment. Verifier time is the verification time each node takes to verify its fragment.

As reported in Table V, when paired with our custom multilinear PCS LightLigero, CONDA consistently achieves **a** $3\sim4\times$ **reduction on per-node communication** while maintaining a comparable prover time (within 25% slowdown) compared to the state-of-the-art FRIDA. This improvement has tangible practical impact: it preserves FRIDA's stellar prover speed while significantly reducing one of its main adoption hurdles—its high communication cost from large commitment sizes. Meanwhile, when paired with Dory PCS [31], CONDA demonstrates a more significant $\sim6\times$ communication reduction from FRIDA, but suffers from a $\sim3\times$ slower prover. The prover time and communication tradeoff between our two instantiations directly reflects the choice of the PCS: Dory has a logarithmic proof size without the large blow-up of a constant factor, but its prover computation involves more expensive group operations. This tradeoff again reflects the flexibility of our commitment-agnostic consolidation scheme which underlies our CONDA.

Secondly, we compare across all code commitment schemes in Table V and visualize the two main efficiency metrics

Code Commitment Scheme	Prover	Per-node Comm.	Verifier
NNT[8]	18.61 s	160.01 KB	33 ms
ADVZ [14]	30.62 s	192.63 KB	11 ms
FRIDA [15]	4.92 s	864.02 KB	495 ms
ZODA [11]	11.58 s	567.88 KB	213 ms
CONDA + Dory	17.16 s	132.08 KB	8 ms
CONDA + LightLigero	5.952 s	196.77 KB	22 ms

TABLE V: Efficiency comparisons among all major code commitment schemes at block size $|M| = 64$MB, total number of nodes $N = 2048$. Single-threaded prover time and per-node communication definition is consistent with Table IV.

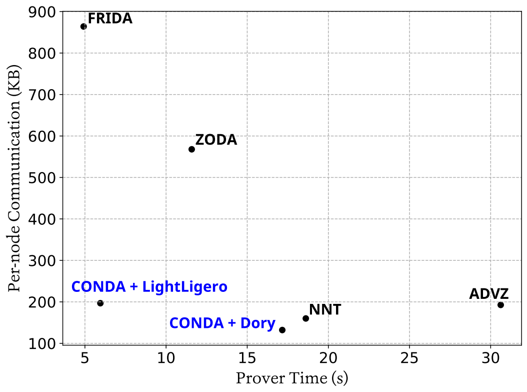

Fig. 3: Cost profile visualized for Table V, lower left is better with faster prover and lower per-node communication.

in Figure 3. Schemes with prover workloads dominated by field operations (e.g., ZODA) or hash-based computations (e.g., the IOPP-based FRIDA) demonstrate superior prover performance relative to KZG-based schemes (e.g., NNT, ADVZ), but at the cost of substantial communication overhead. Our proposed schemes achieve a favorable trade-off, offering faster prover times than the former and significantly improved bandwidth efficiency compared to the latter. Notably, the CONDA + LightLigero construction **simultaneously attains near-best prover efficiency and near-best communication cost**.

B. Multilinear Polynomial Commitment

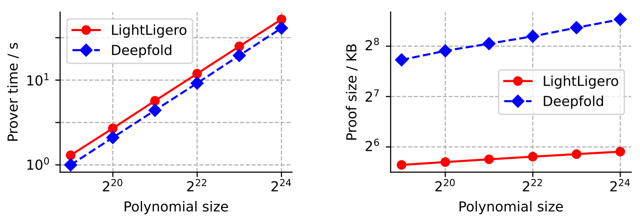

Fig. 4: Single-threaded prover time (left) and per-share proof size (right) of PCS schemes.

To showcase the value of our designed multilinear PCS, we compare the performance of LightLigero with that of Deepfold [35] (Sec'25). Both schemes are based on Reed-Solomon codes, and has similar asymptotic prover time. For all benchmarks, we set the code rate to $1/4$.

We present the concrete performance results in Figure 4. The results show that the prover time of LightLigero is approximately 30% slower than that of Deepfold, but the proof size of LightLigero is at least $4\times$ smaller due to its better asymptotic complexity. For example, for a polynomial of size 2^{21} (corresponding to 64MB of data), the proof size of Deepfold is 265KB, while that of LightLigero is only 52KB. In the DA application discussed in the previous section, most settings involve only tens to hundreds of symbols. In such cases, the proof size of Deepfold introduces a significant overhead to the total communication cost.

C. Verifiable Secret Sharing

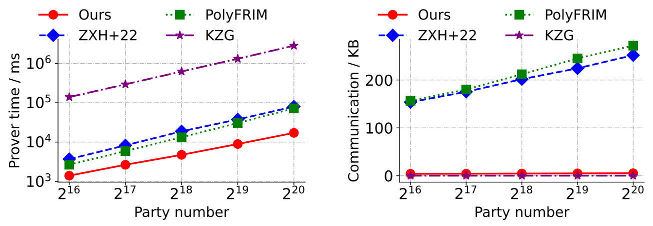

Fig. 5: Single-threaded prover time (left) and per-share communication cost (right) of VSS schemes.

Besides data availability, our RS code commitment can also be applied to verifiable secret sharing (VSS). We benchmark several large-scale VSS schemes against our RS code commitment combined with Dory. We compare our approach with ZXH+22 [23] (Sec'22), PolyFRIM [25] (Sec'24), and amortized KZG multi-evaluation [22], across various party sizes n ranging from 2^{16} to 2^{20}.

Concrete performance results are presented in Figure 5. Our scheme achieves best-in-class prover time among all evaluated solutions. For $n = 2^{20}$, the single-threaded prover time is only 17 seconds, with a communication cost of 5.2KB per symbol. This represents at least an $80\times$ reduction in communication and a $4\times$ improvement in prover time compared to ZXH+22 (using the Virgo [40] PCS)[8] and PolyFRIM.

The MultiEval-based KZG achieves an optimal $O(1)$ communication cost of only 96B, but its prover time exceeds 2800 seconds for $n = 2^{20}$—over $160\times$ slower than our scheme. Moreover, KZG-based solutions require a trusted setup, while our scheme only requires a transparent setup.

References

[1] V. Nikolaenko and D. Boneh, "Data availability sampling and danksharding: An overview and a proposal for improvements," https://a16zcrypto.com/posts/article/an-overview-of-danksharding-and-a-proposal-for-improvement-of-das/, 2024, accessed: 2024-12-28.

[8]While ZXH+22's communication could theoretically be reduced by employing the Dory PCS, integrating it is non-trivial. Even with such integration, the communication cost would still be at least $3\times$ larger than ours, and the prover time would increase.

[2] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, “Narwhal and tusk: a dag-based mempool and efficient bft consensus,” in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 34–50. [Online]. Available: https://doi.org/10.1145/3492321.3519594

[3] M. O. Rabin, *The information dispersal algorithm and its applications*. Berlin, Heidelberg: Springer-Verlag, 1990, p. 406–419.

[4] J. A. Garay, R. Gennaro, C. Jutla, and T. Rabin, “Secure distributed storage and retrieval,” in *Distributed Algorithms*, M. Mavronicolas and P. Tsigas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 275–289.

[5] M. Al-Bassam, A. Sonnino, V. Buterin, and I. Khoffi, “Fraud and data availability proofs: Detecting invalid blocks in light clients,” in *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*. Berlin, Heidelberg: Springer-Verlag, 2021, pp. 279–298.

[6] L. Yang, S. J. Park, M. Alizadeh, S. Kannan, and D. Tse, “DispersedLedger: High-Throughput byzantine consensus on variable bandwidth networks,” in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, 2022, pp. 493–512. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/yang

[7] J. S. Plank, “A tutorial on reed–solomon coding for fault-tolerance in raid-like systems,” *Software: Practice and Experience*, vol. 27, no. 9, pp. 995–1012, 1997.

[8] K. Nazirkhanova, J. Neu, and D. Tse, “Information dispersal with provable retrievability for rollups,” in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, ser. AFT ’22. New York, NY, USA: Association for Computing Machinery, 2023, p. 180–197. [Online]. Available: https://doi.org/10.1145/3558535.3559778

[9] J. Bearer, B. Bünz, P. Camacho, B. Chen, E. Davidson, B. Fisch, B. Fish, G. Gutoski, F. Krell, C. Lin, D. Malkhi, K. Nayak, K. Shen, A. Xiong, N. Yospe, and S. Long, “The espresso sequencing network: HotShot consensus, tiramisu data-availability, and builder-exchange,” Cryptology ePrint Archive, Paper 2024/1189, 2024. [Online]. Available: https://eprint.iacr.org/2024/1189

[10] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Fast reed-solomon interactive oracle proofs of proximity,” in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, vol. 107. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 14:1–14:17. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2018.14

[11] A. Evans, N. Mohnblatt, and G. Angeris, “ZODA: Zero-overhead data availability,” Cryptology ePrint Archive, Paper 2025/034, 2025. [Online]. Available: https://eprint.iacr.org/2025/034

[12] I. S. Reed, G. Solomon, and K. H. March, “Polynomial codes over certain finite fields,” *Journal of The Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[13] M. Hall-Andersen, M. Simkin, and B. Wagner, “Foundations of data availability sampling,” Cryptology ePrint Archive, Paper 2023/1079, 2023. [Online]. Available: https://eprint.iacr.org/2023/1079

[14] N. Alhaddad, S. Duan, M. Varia, and H. Zhang, “Succinct erasure coding proof systems,” 2021, https://eprint.iacr.org/2021/1500.

[15] M. Hall-Andersen, M. Simkin, and B. Wagner, “Frida: Data availability sampling from fri,” in *Annual International Cryptology Conference*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 289–324.

[16] K. D. Bowers, A. Juels, and A. Oprea, “Proofs of retrievability: theory and implementation,” in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 43–54. [Online]. Available: https://doi.org/10.1145/1655008.1655015

[17] C. Cachin and S. Tessaro, “Asynchronous verifiable information dispersal,” *24th IEEE Symposium on Reliable Distributed Systems (SRDS’05)*, pp. 191–201, 2005.

[18] J. Hendricks, G. R. Ganger, and M. K. Reiter, “Verifying distributed erasure-coded data,” in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1281100.1281122

[19] N. Alhaddad, S. Das, S. Duan, L. Ren, M. Varia, Z. Xiang, and H. Zhang, “Asynchronous verifiable information dispersal with near-optimal communication,” Cryptology ePrint Archive, Paper 2022/775, 2022. [Online]. Available: https://eprint.iacr.org/2022/775

[20] A. Kate, G. M. Zaverucha, and I. Goldberg, “Constant-size commitments to polynomials and their applications,” in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–194.

[21] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in *Annual international cryptology conference*. Springer, 1991, pp. 129–140.

[22] D. Feist and D. Khovratovich, “Fast amortized KZG proofs,” Cryptology ePrint Archive, Paper 2023/033, 2023. [Online]. Available: https://eprint.iacr.org/2023/033

[23] J. Zhang, T. Xie, T. Hoang, E. Shi, and Y. Zhang, “Polynomial commitment with a One-to-Many prover and applications,” in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, 2022, pp. 2965–2982. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/zhang-jiaheng

[24] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, “Ligero: Lightweight sublinear arguments without a trusted setup,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2087–2104. [Online]. Available: https://doi.org/10.1145/3133956.3134104

[25] Z. Zhang, W. Li, Y. Guo, K. Shi, S. S. M. Chow, X. Liu, and J. Dong, “Fast RS-IOP multivariate polynomial commitments and verifiable secret sharing,” in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, 2024, pp. 3187–3204. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-zongyang

[26] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[27] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, “Verifiable secret sharing and achieving simultaneity in the presence of faults,” in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, ser. SFCS ’85. USA: IEEE Computer Society, 1985, p. 383–395. [Online]. Available: https://doi.org/10.1109/SFCS.1985.64

[28] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich, “Aggregatable subvector commitments for stateless cryptocurrencies,” in *Security and Cryptography for Networks*, C. Galdi and V. Kolesnikov, Eds. Springer International Publishing, 2020.

[29] A. Tomescu, “How to compute all pointproofs,” Cryptology ePrint Archive, Paper 2020/1516, 2020. [Online]. Available: https://eprint.iacr.org/2020/1516

[30] C. Papamanthou, E. Shi, and R. Tamassia, “Signatures of correct computation,” in *Theory of Cryptography*, A. Sahai, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 222–242.

[31] J. Lee, “Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments,” in *Theory of Cryptography Conference*. Berlin, Heidelberg: Springer-Verlag, 2021, pp. 1–34.

[32] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, “Doubly-efficient zksnarks without trusted setup,” in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 926–943.

[33] T. Xie, Y. Zhang, and D. Song, “Orion: Zero knowledge proof with linear prover time,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 299–328.

[34] R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang, “Ligero++: A new optimized sublinear iop,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 2025–2038.

[35] Y. Guo, X. Liu, K. Huang, W. Qu, T. Tao, and J. Zhang, “DeepFold: Efficient multilinear polynomial commitment from reed-solomon code and its application to zero-knowledge proofs,” Cryptology ePrint Archive, Paper 2024/1595, 2024. [Online]. Available: https://eprint.iacr.org/2024/1595

[36] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[37] H. Zeilberger, B. Chen, and B. Fisch, “Basefold: efficient field-agnostic polynomial commitment schemes from foldable codes,” in *Annual International Cryptology Conference*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 138–169.

[38] A. Kothapalli and S. Setty, “Hypernova: Recursive arguments for customizable constraint systems,” in *Annual International Cryptology Conference*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 345–379.

[39] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *J. ACM*, vol. 39, no. 4, p. 859–868, Oct. 1992. [Online]. Available: https://doi.org/10.1145/146585.146605

[40] J. Zhang, T. Xie, Y. Zhang, and D. Song, "Transparent polynomial delegation and its applications to zero knowledge proof," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 859–876.

[41] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*. Springer, Cham, 2019, pp. 733–764.

[42] A. Kothapalli and B. Parno, "Algebraic reductions of knowledge," in *Annual International Cryptology Conference*. Springer, 2023, pp. 669–701.

APPENDIX A
SECURITY PROPERTIES OF CODE COMMITMENT

A security code commitment should satisfy the following completeness property: For every $\mathsf{gp} \in \mathsf{Setup}(1^\lambda)$, every $m \in \mathbb{F}^{L\times k}$, and every $i \in [n]$, we have

$$\Pr\left[\begin{array}{r|l} \mathsf{Verify}(\mathsf{ck}, \mathsf{com}, & (\mathsf{com}, St) \leftarrow \mathsf{Commit}(\mathsf{ck}, m) \\ i, \hat{m}_i, \tau) = 1 & \hat{m} := \mathsf{RS}(m) \\ & \tau \leftarrow \mathsf{Open}(\mathsf{ck}, St, i) \end{array}\right] = 1$$

Most importantly, erasure code commitments should be position-binding and code-binding, as defined next. Informally, position binding ensures that no adversary can open the codeword at one position to two different values. Code-binding states that whatever an adversary opens is consistent with the code, i.e., the objects to which one commits are really codewords.

Definition 3 (Position-Binding)**.** *We say that a RS code commitment* $\mathsf{CC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *is position-binding, if for every PPT algorithm $\mathcal{A}$, the following advantage is negligible:*

$$\Pr\left[\begin{array}{r|r} \hat{m} \neq \hat{m}' \wedge & \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{Verify}(\mathsf{gp}, \mathsf{com}, i, \hat{m}, \tau) = 1 \wedge & (\mathsf{com}, i, \hat{m}, \tau, \hat{m}', \tau') \\ \mathsf{Verify}(\mathsf{gp}, \mathsf{com}, i, \hat{m}', \tau') = 1 & \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right]$$

Definition 4 (Code-Binding)**.** *We say that a RS code commitment* $\mathsf{CC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *is code-binding, if for every PPT algorithm $\mathcal{A}$, the following advantage is negligible:*

$$\Pr\left[\begin{array}{r|r} & \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \neg(\exists c \in \mathsf{RS}^{\equiv L} : \forall i \in I, c_i = \hat{m}_i) \wedge & (\mathsf{com}, (\hat{m}_i, \pi_i)_{i\in I}) \\ \forall i \in I : \mathsf{Verify}(\mathsf{gp}, \mathsf{com}, i, \hat{m}_i, \pi_i) = 1 & \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right]$$

APPENDIX B
SUMCHECK PROTOCOL

The sumcheck protocol is used to prove the sum of a multivariate polynomial $f \in \mathbb{F}^{\leq d}[X_1, \cdots, X_\mu]$ on a boolean hypercube is some public y:

$$\sum_{b_1,\cdots,b_\mu \in \{0,1\}} f(b_1, \cdots, b_\mu) = y$$

We describe the sumcheck protocol in Protocol 4. The proof size and verifier time of the protocol is $d\mu$. When f is product of d multilinear polynomials [41], which is the most common case, the prover time is $d^2 \cdot 2^\mu$. The sumcheck is perfect complete and sound with $\epsilon = \frac{d\mu}{|\mathbb{F}|}$.

Protocol 4. Sumcheck Protocol.
Input. $\mathcal{P}$ holds a multivariate $f(X_1, \cdots, X_\mu)$, where each variable is of degree d. $\mathcal{V}$ is given oracle access to f, and purported sum y.

1) The i-th round, $i \in \{1, \cdots, \mu\}$:
   a) $\mathcal{P}$ sends polynomial $f_i(X) \in \mathbb{F}^{(\leq d)}[X]$ to $\mathcal{V}$, where f_i is defined as follows:
   $$f_i(X) := \sum_{\vec{b}_{[i+1:]} \in \{0,1\}^{\mu-i}} f\left(\vec{r}_{[:i-1]}, X, \vec{b}_{[i+1:]}\right)$$
   b) $\mathcal{V}$ checks $f_i(0)+f_i(1) = y$ when $i = 1$; otherwise $\mathcal{V}$ checks $f_i(0) + f_i(1) = f_{i-1}(r_{i-1})$.
   c) $\mathcal{V}$ samples $r_i \in \mathbb{F}$ and sends r_i to $\mathcal{P}$.
2) $\mathcal{V}$ queries oracle of f to obtain $f(\vec{r})$ and checks $f_\mu(r_\mu) = f(\vec{r})$. $\mathcal{V}$ outputs 1 if all the checks pass, otherwise outputs 0.

APPENDIX C
EVALUATION REDUCTION DEFINITION

We define the definition of *evaluation reduction* based on *reduction of knowledge* in [42]. First, we define an evaluation relation $\mathcal{R}_{\mathsf{Eval}}$ as follows:

$$\left\{\left(\mathcal{C}_f, \vec{z}, y; \tilde{f}\right) : \tilde{f}(\vec{z}) = y \wedge \mathcal{C}_f = \mathsf{mlPC.Commit}(\tilde{f})\right\}$$

where $\tilde{f}$ is a μ-variate multilinear polynomial, $\vec{z} \in \mathbb{F}^\mu, y \in \mathbb{F}$. Evaluation reduction is a public-coin interactive protocol between $\mathcal{P}$ and $\mathcal{V}$, denoted as

$$(\vec{r}, y_r) \leftarrow \langle \mathcal{P}(\tilde{f}), \mathcal{V}\rangle(\mathcal{C}_f, \vec{z}, y)$$

where $\vec{r} \in \mathbb{F}^\mu, y_r \in \mathbb{F}$. The evaluation reduction protocol satisfies the following properties:

- Completeness: for any $\left(\mathcal{C}_f, \vec{z}, y; \tilde{f}\right) \in \mathcal{R}_{\mathsf{Eval}}$, the following probability is 1:
$$\Pr\left[(\vec{r}, y_r) \leftarrow \langle \mathcal{P}(\tilde{f}), \mathcal{V}\rangle(\mathcal{C}_f, \vec{z}, y) : \left(\mathcal{C}_f, \vec{r}, y_r; \tilde{f}\right) \in \mathcal{R}_{\mathsf{Eval}}\right]$$
- Soundness: for any PPT prover $\mathcal{P}^*$, the following probability is *negligible*:
$$\Pr\left[\begin{array}{c} (\vec{r}, y_r) \leftarrow \langle \mathcal{P}(\tilde{f}), \mathcal{V}\rangle(\mathcal{C}_f, \vec{z}, y) \\ \wedge \left(\mathcal{C}_f, \vec{r}, y_r; \tilde{f}\right) \in \mathcal{R}_{\mathsf{Eval}} \end{array} : \left(\mathcal{C}_f, \vec{z}, y; \tilde{f}\right) \notin \mathcal{R}_{\mathsf{Eval}}\right]$$

Informally, the soundness states that if the original claim of $\tilde{f}(\vec{z}) = y$ isn't correct, with overwhelming probability, the reduced evaluation $f(\vec{r}) \neq y_r$.

APPENDIX D
INTERLEAVED RS CODE COMMITMENT FULL SCHEME

The full protocol of our interleaved RS code commitment scheme is presented in Protocol 5

Protocol 5. CONDA: **Interleaved RS Code Commitment Scheme**
Setup: $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given the security parameter and output $\mathsf{pp} := (\mathsf{mlPC.Setup}(1^\lambda), \Omega, s, n)$.
Prove: $\mathsf{Prove}(\mathsf{pp}, M) \to (\mathcal{C}, \{\pi_j\}_{j\in[n]})$

1) Parses the input $M \in \mathbb{F}^{L\times k}$ as a bivariate polynomial $f(X,Y)$ such that $f(X,Y) = \sum_{i\in[L],j\in[k]} M[i,j]\cdot X^iY^j$. Let $\tilde{f}$ be the twin polynomial of $f(X,Y)$, compute $\mathcal{C} \leftarrow \mathsf{mlPC.Commit}(\mathsf{pp}, \tilde{f})$.
2) Encode M into interleaved RS codeword $\vec{c} = (\vec{c}^{(0)}, \cdots, \vec{c}^{(n-1)}) \in (\mathbb{F}^L)^n$ and constructs vector $\vec{v}$ of length n, such that its i-th element is a univariate polynomial $v_j(X) = \sum_{i\in[L]} \vec{c}^{(j)}[i]\cdot X^i$ whose coefficient is $\vec{c}^{(j)}$.
3) Merkle-commit all $v_j(X)$ to derive the partial evaluation point: $mt \leftarrow \mathsf{MT.Construct}(\vec{v})$, $\mathsf{FS.add}(mt.\mathsf{Root}\|\mathcal{C}), \beta \leftarrow \mathsf{FS.chal}()$; Prepare merkle path for every $\forall j\in[n]$, $\pi_{mt,j} \leftarrow \mathsf{MT.Open}(j)$.
4) Consolidate all partial evaluations: $(\vec{r}, \{\vec{\pi}_{\mathsf{EC},j}\}) \leftarrow \mathsf{NIEC.Consolidate}(\vec{e}, \Omega)$ where $\vec{e}[j] = v_j(\beta)$.
5) Run evaluation on the consolidated point: $(y_r, \pi_{\mathsf{pc}}) \leftarrow \mathsf{MLPC.Eval}(\mathsf{pp}, \vec{r}, \tilde{f})$.
6) Outputs $\mathcal{C}$ and $\{\pi_j := (\pi_{\mathsf{EC},j}, \pi_{mt,j}, \pi_{\mathsf{pc}}, \vec{r}, y_r)\}_{j\in[n]}$.

Verify: $\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, j, v_j(X), \pi) \to b$

1) Parse $(\pi_{\mathsf{EC}}, \pi_{mt}, \pi_{\mathsf{pc}}) \leftarrow \pi$, verify the merkle proof $rt \leftarrow \mathsf{MT.Recover}(\pi_{mt}, j)$, and derive the partial evaluation point $\mathsf{FS.add}(rt\|\mathcal{C}), \beta \leftarrow \mathsf{FS.chal}()$.
2) Outputs $\mathsf{NIEC.Verify}(\vec{z}_j, v_j(\beta), \vec{r}, \pi_{\mathsf{EC}}) \wedge \mathsf{mlPC.Verify}(\mathcal{C}, \vec{r}, y_r, \pi_{\mathsf{pc}})$ where $\vec{z}_j = (\omega^j, \omega^{2j}, \ldots, \omega^{2^{\mu-1}\cdot j})$.

Appendix E
Proof of Theorems

A. Theorem 2

Proof. Define $f_i = p_i(r_i)$, where $p_i(X)$ is the univariate polynomial sent to $\mathcal{V}$ in the i-th round. For simplicity, we assume s is divisible by μ. Define $d = \frac{\mu}{s}$ and $f_0 = y$. For any $i \in [1,d]$,

$$\begin{aligned}
&\Pr\left[\tilde{f}(\vec{r}_{[:i\cdot s]}, \vec{z}_{[i\cdot s+1:]}) = f_i\right]\\
\le &\Pr\left[\begin{array}{c|c} \tilde{f}(\vec{r}_{[:i\cdot \ell]}, \vec{z}_{[i\cdot s+1:]}) & \tilde{f}(\vec{r}_{[:(i-1)s]}, X^{2^0}, \cdots, X^{2^{s-1}}, \vec{z}_{[i\cdot s+1:]}) \\ = p_i(r_i) & \neq p_i(X)\end{array}\right]\\
&+ \Pr[\tilde{f}(\vec{r}_{[:(i-1)\cdot s]}, X^{2^0}, \cdots, X^{2^{s-1}}, \vec{z}_{[i\cdot s+1:]}) = p_i(X)]\\
\le &\frac{2^s}{|\mathbb{F}|} + \Pr[\tilde{f}(\vec{r}_{[:(i-1)\cdot s]}, \vec{z}_{[(i-1)\cdot s+1:]}) = f_{i-1}]
\end{aligned}$$

Thus we have

$$\begin{aligned}
&\Pr[\tilde{f}(\vec{r}_{[:i\cdot s]}, \vec{z}_{[i\cdot s+1:]}) = f_i]\\
&\quad - \Pr[\tilde{f}(\vec{r}_{[:(i-1)\cdot s]}, \vec{z}_{[(i-1)s+1:]}) = f_{i-1}] \le \frac{2^s}{|\mathbb{F}|}
\end{aligned} \tag{1}$$

For $i \in [\mu]$, we sum up (1) and notice that $\Pr[\tilde{f}(\vec{z}) = f_0] = 0$.

$$\Pr[y_r = \tilde{f}(\vec{r})] = \Pr[y_r = \tilde{f}(\vec{r})] - \Pr[\tilde{f}(\vec{z}) = f_0] \le \frac{\mu\cdot 2^s}{s\cdot|\mathbb{F}|}$$

□

B. Theorem 3

Proof. We first assume PST, sumcheck and Merkle tree has perfect soundness.

We call a vector $\vec{v}$ *approximates* RS code RS, if and only if there exists codeword $\vec{c} \in \mathsf{RS}$ such that $\Delta(\vec{v}, \vec{c}) < 1-\rho$, where $\Delta(\cdot,\cdot)$ represents relative hamming distance of two vectors. Let $\vec{v}'$ be the vector folded from the committed matrix C. Let $\vec{f}'$ be the codeword corresponding to the PST committed multilinear polynomial. If $\vec{v}'$ doesn't approximate $\vec{f}'$, which means $\Delta(\vec{v}', \vec{f}') \ge 1-\rho$, then for each random sample $x \in [2^{\mu-\mu'}]$,

$$\Pr\left[\vec{v}'[x] = \vec{f}'[x]\right] \le \rho$$

Then for s independent random samples,

$$\Pr\left[\forall x\in I, \vec{v}'[x] = \vec{f}'[x]\right] \le \rho^s$$

Consider the negligible soundness error from Merkle tree and PST batched open, the overall soundness error is $\rho^s + \mathsf{negl}(\lambda)$. □

Appendix F
Verifiable Secret Sharing Scheme

We list how to construct VSS scheme from our scheme. The only challenge is ensuring the multi-evaluation proof zero-knowledge. We need an additive homomorphic zk-multilinear PCS. A multilinear PCS with zero-knowledge property is defined as follows:

Definition 5. *A polynomial commitment scheme is zero-knowledge if there is a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all multilinear polynomial $\tilde{f}$, PPT non-uniform adversary $\mathcal{A}$:*

$$\Pr\left[b=1\ \middle|\ \begin{array}{r} \mathcal{C}\leftarrow Commit(pp,\tilde{f})\\ \vec{z}\leftarrow\mathcal{A}(\mathcal{C},pp)\\ (y,tr)\leftarrow\langle\mathcal{P}(\tilde{f}),\mathcal{V}\rangle(\vec{z})\\ b\leftarrow\mathcal{A}(\mathcal{C},y,\vec{z},tr)\end{array}\right] =$$

$$\Pr\left[b=1\ \middle|\ \begin{array}{r} \mathcal{C}\leftarrow\mathcal{S}_1(pp)\\ \vec{z}\leftarrow\mathcal{A}(\mathcal{C},pp)\\ (y,tr)\leftarrow\mathcal{S}_2(\tilde{f}(\vec{z}),\vec{z})\\ b\leftarrow\mathcal{A}(\mathcal{C},y,\vec{z},tr)\end{array}\right]$$

We present a construction of mKZG, which is a candidate of homomorphic zk-multilinear PCS.

- $\mathsf{Setup}(1^\lambda, \mu) \to \mathsf{pp}$: Sample random $\tau_0, \cdots, \tau_\mu \in \mathbb{F}$. Output $\mathsf{pp} = \left(\tau_0 \cdot g, \{\prod_{i \in W} \tau_i \cdot g\}_{W \subset [1,\mu]}\right)$.
- $\mathsf{Commit}(\mathsf{pp}, \tilde{f}) \to (\mathcal{C}, \mathcal{D})$: Sample $r \in \mathbb{F}$. Compute $\mathcal{C} = r \cdot (\tau_0 \cdot g) + \tilde{f}(\tau_1, \cdots, \tau_\mu) \cdot g$.
- $\mathsf{Eval}(\mathsf{pp}, \mathcal{D}, \vec{z}, \tilde{f}) \to (y, \pi)$: Sample $r_0, \cdots, r_\mu \in \mathbb{F}$, compute $c = \sum_{i \in [\mu]} r_i \cdot (\tau_i \cdot g)$. Compute $y = \tilde{f}(\vec{z}), y_r = \sum r_i \cdot z_i$. Compute $\alpha = \mathsf{Hash}(\mathcal{C}||c||\vec{z}||y||y_r)$. Define $\tilde{f}_r(X_1, \cdots, X_\mu) = \tilde{f} + \alpha \sum r_i \cdot X_i$. Define
$$\tilde{Q}_i(X_{i+1}, \cdots, X_\mu) := \frac{\tilde{f}_r(z_{[:i-1]}, X_{[i]}) - \tilde{f}_r(z_{[:i]}, X_{[i+1:]})}{X_i - z_i}$$
for each $i \in [1, \mu]$. Compute $\beta = r + \alpha r_0$, $\pi_i = \tilde{Q}_i(\tau_{i+1}, \cdots, \tau_\mu)$. Define $\pi = \left(c, y_r, \beta, \{\pi_i\}_{i \in [\mu]}\right)$
- $\mathsf{Verify}(\mathsf{pp}, \mathcal{C}, \vec{z}, y, \pi) \to 0/1$. Parse $\pi = \left(c, y_r, \beta, \{\pi_i\}_{i \in [\mu]}\right)$. Compute $\alpha = \mathsf{Hash}(\mathcal{C}||c||\vec{z}||y||y_r)$. Check $\prod_{i \in [1,\mu]} e\left(\pi_i, (\tau_i - z_i) \cdot g\right) = e\left(\mathcal{C} + \alpha c - (\beta\tau_0 + y + \alpha y_r) \cdot g, g\right)$.

The commitment is additive homomorphic, since given commitment of two multilinear polynomials $\tilde{f}_1, \tilde{f}_2$, anyone can derive the commitment of $\tilde{f}_1 + \tilde{f}_2$. Armed with zk-mlPC, we present the construction of our VSS in Protocol 6.

Protocol 6. Zero-knowledge Evaluation Consolidation

Setup: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: It takes the security parameter and output $\mathsf{pp} = \mathsf{mlPC.Setup}(1^\lambda)$

Deal $(\mathsf{com}, St) \leftarrow \mathsf{Deal}(\mathsf{pp}, s)$:

1) $\mathcal{P}$ generates random polynomial $f(X)$ of degree $k-1$ such that $f(0) = s$. $\mathcal{P}$ generates another random polynomial $g(X)$ of degree $k-1$.
2) Let $\tilde{f}, \tilde{g}$ be the twin polynomial of $f(X)$ and $g(X)$. $\mathcal{P}$ invokes $(\mathcal{C}_f, \mathcal{D}_f) \leftarrow \mathsf{mlPC.Commit}(\mathsf{ck}, \tilde{f})$, $(\mathcal{C}_g, \mathcal{D}_g) \leftarrow \mathsf{mlPC.Commit}(\mathsf{ck}, \tilde{g})$.
3) $\mathcal{P}$ constructs vector $\vec{p}$ of length n, such that its i-th element is a linear function $p_i(X) = f(\omega^i) + g(\omega^i) \cdot X$, where ω is n-th root of unity.
4) $\mathcal{P}$ invokes $mt^{(0)} \leftarrow \mathsf{MT.Construct}(\vec{p})$. $\beta \leftarrow \rho\left(mt^{(0)}||\mathcal{C}_f||\mathcal{C}_g\right)$, $(\vec{r}, y, \vec{mt}, p(X)) \leftarrow \mathsf{EC}(\vec{c})$, where $\vec{c} = (p_0(\beta), \cdots, p_{n-1}(\beta))$.
5) $\mathcal{P}$ invokes $(y, \pi_{\mathsf{pc}}) \leftarrow \mathsf{mlPC.Eval}(\mathsf{ck}, (\mathcal{D}_f, \mathcal{D}_g), \vec{z}, \tilde{f} + \beta \cdot \tilde{g})$.
6) Let $\mathsf{com} = (\mathcal{C}_f, \mathcal{C}_g, \pi_{\mathsf{pc}})$, $St = (mt^{(0)}, \vec{mt}, p(X))$.
7) For $\forall i \in [n]$, run $\pi_i \leftarrow \mathsf{Open}(\mathsf{ck}, St, i)$ and send π_i to the i-th party.

Open: $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, St, i)$:

1) Parse St as vector of Merkle trees $mt^{(0)}$, $\vec{mt}$ and univariate polynomial $p(X)$.
2) Invoke $\mathsf{MT.Open}$ to open the i-th path of $mt^{(0)}$, denote the result as $path_0$.
3) For each $mt \in \vec{mt}$, let d be the leaf number of mt, invoke $\mathsf{MT.Open}$ to open its $(i \bmod d)$-th leaf and path. Put all these leaves and paths into a list $\vec{u}$. Output $\pi = (path_0, \vec{u}, p(X))$.

Verify: $\mathsf{Verify}(\mathsf{ck}, \mathsf{com}, i, v(X), \pi) \rightarrow b$

1) Parse $\mathsf{com} = (\mathcal{C}_f, \mathcal{C}_g, \pi_{\mathsf{pc}})$, parse π as $path_0$, $\vec{u}$ and $p(X)$.
2) Invoke $\beta \leftarrow \rho\left(\mathsf{MT.Recover}(path_0, p_i, i)||\mathcal{C}_f||\mathcal{C}_g\right)$.
3) Invoke $(\vec{r}, y_r) \leftarrow \mathsf{vRed}(p_i(\beta), \vec{u}, i, p)$.
4) Output $\mathsf{mlPC.Verify}(\mathcal{C}_f + \beta \cdot \mathcal{C}_g, \vec{r}, y_r, \pi_{\mathsf{pc}})$