Lightning: A Fast Field-Agnostic Polynomial Commitment Scheme

Wenjie Qu, Yanpei Guo, and Jiaheng Zhang

National University of Singapore

Abstract. Polynomial commitment schemes (PCS) are a fundamental building block of modern zkSNARKs. In this paper, we propose *Lightning*, a new coding-based PCS that achieves state-of-the-art prover efficiency. Our main technical contribution is a systematic code family, the *Lightning code*, which can be instantiated from any base code with constant relative distance. Compared with the base code, the Lightning code significantly reduces encoding cost by trading off relative distance. At a comparably small relative distance, Lightning's $\delta = 0.005$ configuration has an equivalent encoding cost 2.8× lower than Brakedown-fastest at $\delta = 0.0043$; at the common relative distance $\delta = 0.020$, its equivalent encoding cost is 1.53× lower than that of the Brakedown code. We integrate the Lightning code into the standard coding-based PCS framework of Ligero and Brakedown. When committing to 2^{28} coefficients at matched proof sizes, Lightning reduces commitment time from about 349s to 256s (1.36× faster) for 80MB proofs, compared with Brakedown.

Keywords: Polynomial commitment schemes · Error-correcting codes · zkSNARKs

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) enable a prover to convince a verifier of the correctness of a computation without revealing any additional information. Their succinctness and efficient verification make them particularly attractive for various applications, including blockchain rollups [27,4], decentralized identity [28], anonymous payment [32], and verifiable machine learning inference [26].

Modern zkSNARK constructions [19,12] largely follow the now-standard Polynomial Commitment Scheme (PCS) [21,35]+ Polynomial Interactive Oracle Proof (PIOP) [31,18,33] paradigm. A PCS enables the prover to commit to a μ-variate polynomial f defined over a field $\mathbb{F}$ using a short commitment. Given $\overrightarrow{x} \in \mathbb{F}^{\mu}$, the prover can create a proof to convince the verifier that $f(\overrightarrow{x}) = y$ for some $y \in \mathbb{F}$. Among these components, the PCS is often the dominant cost for the prover.

Since the PCS often dominates prover time, improving PCS efficiency has long been a central research direction in the study of zkSNARKs [38,37,21]. Over the past decade, multiple PCS designs have been proposed. The concept

and initial construction of univariate PCS (namely, $\mu = 1$) originated with the pioneering KZG construction [24], which is based on the discrete-logarithm assumption and pairing groups. PST [30] extended the PCS concept to multilinear polynomials ($\mu > 1$, with degree at most one in each variable). However, KZG and PST require a trusted setup and incur expensive group operations. In contrast, many recent PCS constructions are based on error-correcting codes, including Brakedown [21], Orion [36], Virgo [38], BaseFold [37], and WHIR [3]. These schemes have attracted significant attention due to their concrete efficiency, transparent setup, and plausible post-quantum security.

	Prover Time	Proof Size	Verifier Time	Field Agnostic
Ligero (CCS'17) [2]	$O(N \log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	✗
Virgo (S&P'20) [38]	$O(N \log N)$	$O(\log^2 N)$	$O(\log^2 N)$	✗
Orion (CRYPTO'22) [36]	$O(N)$	$O(\log^2 N)$	$O(\log^2 N)$	✗
Brakedown (CRYPTO'23) [21]	$O(N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	✓
BaseFold (CRYPTO'24) [37]	$O(N \log N)$	$O(\log^2 N)$	$O(\log^2 N)$	✓
EA (CRYPTO'24) [6]	$O(N \log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	✓
WHIR (EUROCRYPT'25) [3]	$O(N \log N)$	$O(\log^2 N)$	$O(\log^2 N)$	✗
Ours	$O(N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	✓

Table 1. Performance of coding-based multilinear PCS. N is the size of the polynomial.

1.1 Our Contribution

In this paper, we propose *Lightning*, a new PCS that achieves state-of-the-art prover efficiency and outperforms all existing constructions in terms of commitment cost. Our design follows the classical coding-based PCS framework of Ligero [2] and Brakedown [21], which constructs PCS from linear error-correcting codes and Merkle trees. Within this framework, any linear code with constant relative distance can be generically compiled into a PCS, and the overall prover complexity is dominated by the encoding cost of the underlying code. This abstraction highlights encoding efficiency as the central bottleneck in improving prover performance.

To reduce this encoding bottleneck, we introduce a new linear code family, the *Lightning code*, which can be instantiated from any base code with constant relative distance. The Lightning code achieves substantially lower encoding cost than its base code while preserving constant relative distance. At a high level, it avoids encoding the entire message using the base code, yet still guarantees constant relative distance.

Concretely, given a message $m \in \mathbb{F}_q^n$, Lightning code first applies an efficiently computable linear map $A : \mathbb{F}_q^n \to \mathbb{F}_q^{\alpha n}$, for some $\alpha < 1$, producing a compressed linear sketch mA. It then encodes only this sketch using a large-distance base code $\mathcal{C}_D$. The resulting Lightning codeword is systematic:

$$\mathcal{C}_L(m) := m \,\|\, \mathcal{C}_D(mA),$$

Table 2. Encoding cost comparison of different linear codes used in zkSNARKs. n denotes the length of the message to be encoded.

		Rate	Distance	Encoding cost		
				$\mathbb{F}_\times$	$\mathbb{F}_+$	Total
Reed-Solomon code	$n = 2^{16}$	0.5	0.5	$16n$	$32n$	$24n$
	$n = 2^{20}$	0.5	0.5	$20n$	$40n$	$30n$
Brakedown code [21]	Parameters from improved analysis in Haböck et al. [23] Table 1	0.704	0.02	$9.9n$	$9.9n$	$12.4n$
		0.680	0.03	$11.7n$	$11.7n$	$14.6n$
		0.657	0.04	$12.8n$	$12.8n$	$16.0n$
		0.610	0.05	$13.7n$	$13.7n$	$17.1n$
		0.619	0.06	$15.8n$	$15.8n$	$19.7n$
		0.581	0.07	$18.3n$	$18.3n$	$22.8n$
	Fastest (this work)	0.904	0.0043	$8.07n$	$8.07n$	$10.09n$
Lightning code		0.881	0.005	$1.13n$	$10.13n$	$3.66n$
		0.795	0.010	$2.16n$	$10.16n$	$4.69n$
		0.733	0.015	$3.56n$	$11.56n$	$6.45n$
		0.667	0.020	$4.88n$	$12.88n$	$8.10n$
		0.581	0.025	$7.04n$	$14.04n$	$10.55n$
		0.560	0.030	$8.33n$	$16.33n$	$12.41n$

• The encoding cost is estimated in equivalent $\mathbb{F}_\times$ operations, assuming $1\ \mathbb{F}_\times = 4\mathbb{F}_+$.
• The final Brakedown row is the minimum found by the grid search based on Haböck et al. [23]; the preceding rows are their pre-existing operating points used as Lightning base codes.

so that the expensive encoder $\mathcal{C}_D$ is applied to an input smaller by a factor of α.

The key technical requirement on the linear map A is *sparsity preservation*: it must retain nontrivial information about sparse messages. Concretely, for every nonzero m with support size at most βn, we require $mA \neq 0$. Under this condition, any nonzero message contributes codeword weight either directly through the first part m, or through the encoded component $\mathcal{C}_D(mA)$. As a result, the Lightning code achieves linear minimum distance.

To enable efficient computation of mA, we construct A as a randomly sampled low-density parity-check (LDPC) matrix—specifically, a row-regular sparse binary matrix. Such matrices admit extremely fast multiplication: computing mA requires only $O(n)$ field additions and no field multiplications. In contrast, encoding with the base code involves extensive field multiplications. Combined with the reduced input size for the base code, this design ensures that encoding with the Lightning code is substantially faster than encoding with its base code.

We instantiate Lightning code using the Brakedown code as the base code $\mathcal{C}_D$ and integrate it into standard coding-based PCS constructions, obtaining the Lightning PCS. At a comparably small relative distance, Lightning at $\delta = 0.005$ has an equivalent encoding cost 2.8× lower than Brakedown-fastest at $\delta = 0.0043$. At the common relative distances $\delta = 0.020$ and $\delta = 0.030$, Lightning has equivalent encoding costs 1.53× and 1.17× lower, respectively, than Brakedown. This encoding advantage translates directly into a better PCS trade-off: in the 2^{28}-coefficient benchmark, at the matched proof sizes of 80MB and 53MB, Lightning reduces Brakedown's commitment time from about 349s to 256s (1.36× faster) and from about 410s to 346s (1.18× faster), respectively. Thus, Lightning retains a concrete commitment-time advantage when proof size is held fixed.

We summarize our contributions as follows:

- We propose Lightning, a field-agnostic multilinear polynomial commitment scheme that substantially reduces prover encoding cost. Lightning achieves linear-time encoding dominated by field additions, and attains significantly lower concrete encoding cost than prior coding-based PCS constructions.
- Our core technical contribution is the Lightning code, a new linear code family that achieves efficient encoding by applying the expensive base code only to a compressed linear sketch of the message, rather than the full message, while still guaranteeing constant relative distance.
- For code-distance analysis, we develop a direct counting bound based on an exact occupancy enumeration. It significantly tightens the classical double union-bound analysis for LDPC matrices in Silver [16], yielding substantially larger concrete distance certificates.
- We implement Lightning within standard coding-based PCS frameworks and benchmark it against existing schemes, including Brakedown, Orion, Ligero, and EA. Our results show that Lightning has an equivalent encoding cost 2.8× lower than Brakedown-fastest at comparably small relative distances, and costs 1.53× and 1.17× less than Brakedown at the common relative distances 0.020 and 0.030. At matched proof sizes for committing to 2^{28} coefficients, it is 1.36× faster to commit than Brakedown at 80MB proofs; Lightning also improves the commitment time by up to 3× compared to Orion, and 14× compared to EA.

1.2 Related Work

In recent years, there has been significant progress in constructing concretely efficient polynomial commitment schemes (PCS). Existing constructions are based on a variety of cryptographic assumptions and techniques, including bilinear maps [24,30,25], discrete logarithms [35], groups of unknown order [11], lattices [1], and error-correcting codes [5,2]. Among these, coding-based constructions are particularly appealing due to their transparent setup, simple algebraic structure, and plausible post-quantum security. Our work focuses on coding-based PCS.

The error-correcting codes adopted by prior PCS schemes can be broadly categorized into three classes: (1) foldable codes [37], (2) Brakedown code [21], and (3) Expander-Accumulation (EA) code [9]. These classes exhibit different trade-offs between encoding complexity and relative distance, which in turn directly affect prover efficiency and proof size in the resulting PCS.

Foldable codes typically achieve the largest relative distance among these three classes, but incur $O(n \log n)$ encoding complexity. This category includes Reed–Solomon codes and randomized foldable codes [37]. Many PCS constructions adopt foldable codes, including Ligero [2], FRI [5], Virgo [38], Basefold [37], and WHIR [3]. Due to this quasilinear encoding cost, these PCS schemes incur $O(N \log N)$ prover commitment time (in field operations), where N is the polynomial size. This overhead becomes a major bottleneck in large-scale settings.

The Brakedown code was introduced by Golovnev et al. [21] by generalizing Spielman's binary code [34] to large fields. Its key advantage is strictly linear encoding complexity in the message size, making it particularly attractive for improving prover efficiency. By adopting the Brakedown code, Brakedown PCS was the first to achieve $O(N)$ prover time (in terms of field operations). Subsequent works focused on improving the proof size: Orion [36] reduced Brakedown's proof size from $O(\sqrt{N})$ to $O(\log^2 N)$ by leveraging proof recursion, but required NTT-friendly fields. Brakingbase [29] removed this restriction, making the construction field-agnostic while retaining Orion's proof size. However, these follow-up works do not improve prover time beyond Brakedown, leaving a gap in further optimizing the encoding cost.

The EA code [9] achieves a larger concrete relative distance than the Brakedown code, at the cost of $O(n \log n)$ encoding complexity. Block et al. [6] designed a PCS based on the EA code, yielding a different trade-off between prover time and proof size compared to prior constructions.

More recently, Blaze [10] introduced an efficient PCS based on Repeat-Accumulate-Accumulate (RAA) codes [17]. However, Blaze is limited to binary fields, while our PCS supports general finite fields.

Concurrent work. Independently, Gurkan et al. [22] introduced Bolt, whose construction similarly pursues fast encoding through sparse-sketching techniques. Bolt focuses on SNARKs over binary fields and does not provide a corresponding code-distance analysis over general $\mathbb{F}_q$. Lightning targets the more general $\mathbb{F}_q$ setting used by most zkSNARKs and establishes a distance guarantee in that setting.

2 Preliminaries

2.1 Multilinear extension

Define the equality function $eq(x, y)$, which takes as input two bit strings x and y of the same length ℓ. The function $eq(\cdot, \cdot)$ returns 1 if $x = y$ and 0 otherwise. Let $\tilde{eq}(\cdot, \cdot)$ denote the multilinear extension of $eq(\cdot, \cdot)$. Then $\tilde{eq}(x, y) = \prod_{i=1}^{\ell}((1-x_i)(1-y_i)+x_i y_i)$. The definition of a multilinear extension is as follows:

Definition 1 (Multilinear Extension [15]). *Let $V : \{0,1\}^\ell \to \mathbb{F}$ be a function. The multilinear extension of V is the unique polynomial $\tilde{V} : \mathbb{F}^\ell \to \mathbb{F}$ such that $\tilde{V}(x_1, x_2, \ldots, x_\ell) = V(x_1, x_2, \ldots, x_\ell)$ for all $x_1, x_2, \ldots, x_\ell \in \{0,1\}^\ell$. It can be expressed as $\tilde{V}(x_1, x_2, \ldots, x_\ell) = \sum_{b \in \{0,1\}^\ell} \tilde{eq}(x, b) \cdot V(b)$.*

For a vector Z of length 2^ℓ, its multilinear extension $\tilde{Z}$ is defined as

$$\tilde{Z}(x_1, \ldots, x_\ell) = \sum_{i \in [2^\ell]} \tilde{eq}(x, \mathsf{bin}(i-1)) \cdot Z_i.$$

2.2 Polynomial commitment

A polynomial commitment scheme consists of three algorithms:

- $\mathsf{PC.Commit}(\phi(\cdot))$: outputs a commitment $\mathcal{C}$ to the polynomial $\phi(\cdot)$.
- $\mathsf{PC.Prove}(\phi, \overrightarrow{x})$: given an evaluation point $\overrightarrow{x}$, outputs a tuple $(\overrightarrow{x}, \phi(\overrightarrow{x}), \pi)$, where π is the proof.
- $\mathsf{PC.VerifyEval}\,(\pi, \overrightarrow{x}, y, \mathcal{C})$: given π, $\overrightarrow{x}$, y, and $\mathcal{C}$, checks whether y is the correct evaluation and outputs accept or reject.

Definition 2 ((Multivariate) Polynomial commitment scheme). *A polynomial commitment scheme has the following properties:*

- *Correctness. For every polynomial ϕ and evaluation point $\overrightarrow{x}$, the following probability holds:*

$$\Pr\left(\begin{array}{r} \mathsf{PC.Commit}(\phi) \to \mathcal{C} \\ \mathsf{PC.Prove}(\phi, \overrightarrow{x}) \to \overrightarrow{x}, y, \pi \\ y = \phi(\overrightarrow{x}) \\ \mathsf{PC.VerifyEval}(\pi, \overrightarrow{x}, y, \mathcal{C}) \to \mathsf{accept} \end{array}\right) = 1$$

- *Knowledge Soundness. For any PPT adversary $\mathcal{P}^*$ equipped with algorithms $\mathsf{PC.Commit}^*$ and $\mathsf{PC.Prove}^*$, there exists a PPT extractor $\mathcal{E}$ such that the probability below is negligible:*

$$\Pr\left(\begin{array}{c} \mathsf{PC.Commit}^*\,(\phi^*) \to \mathcal{C}^* \\ \mathsf{PC.Prove}^*\,(\phi^*, \overrightarrow{x}) \to \overrightarrow{x}, y^*, \pi^* : \phi^* \leftarrow \mathcal{E}\,(\mathcal{C}^*, \overrightarrow{x}, \pi^*, y^*) \wedge y^* \neq \phi^*(\overrightarrow{x}) \\ \mathsf{PC.VerifyEval}\,(\pi^*, \overrightarrow{x}, y^*, \mathcal{C}^*) \to \mathsf{accept} \end{array}\right)$$

2.3 Error-Correcting Codes

Definition 3 (Linear Code). *A linear error-correcting code with message length k and codeword length n is a linear subspace $C \subseteq \mathbb{F}^n$ with an injective encoder $E_C : \mathbb{F}^k \to C$. Any linear combination of codewords is also a codeword. The rate of the code is defined as $\frac{k}{n}$. The distance between two codewords u, v is the Hamming distance denoted by $\Delta(u, v)$. The minimum distance is $d = \min_{u,v} \Delta(u, v)$. Such a code is denoted as an $[n, k, d]$ linear code, and we further define its relative distance as $\mathsf{rel_dist}(C) := \frac{d}{n}$.*

2.4 Merkle Tree

Let $H : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be a hash function. A Merkle tree is a data structure that allows one to commit to $l = 2^d$ messages using a single hash value h, such that revealing any message bit requires $d + 1$ hash values.

A Merkle tree is represented by a binary tree of depth d, in which the l message elements $m_1, m_2, \ldots, m_l$ are assigned to the leaves. The values at internal nodes are computed by hashing the values of their two children. To reveal m_i, one reveals m_i together with the values on the path from m_i to the root. We denote the algorithms as follows:

1. $h \leftarrow \mathsf{Merkle.Commit}\,(m_1, \ldots, m_l)$.
2. $(m_i, \pi_i) \leftarrow \mathsf{Merkle.Open}(m, i)$.
3. $\{\mathsf{accept}, \mathsf{reject}\} \leftarrow \mathsf{Merkle.Verify}\,(\pi_i, m_i, h)$.

2.5 Building PCS from Error-Correcting Codes and Tensor IOPs

We next illustrate the coding-based PCS paradigm of [8,21,36,6,2], which builds a PCS from a linear code and tensor queries.

Brakedown [21] observed that a polynomial evaluation can be expressed as a tensor product. Here, we consider only multilinear polynomial commitments, but the paradigm extends directly to univariate polynomials.

In particular, given a multilinear polynomial ϕ, its evaluation at the input vector $x_0, x_1, \ldots, x_{\log N-1}$ is:

$$\phi\left(x_0, \ldots, x_{\log N-1}\right) = \sum_{i_0 \in \{0,1\}} \cdots \sum_{i_{\log N-1} \in \{0,1\}} w_{i_0 i_1 \ldots i_{\log N-1}} x_0^{i_0} \ldots x_{\log N-1}^{i_{\log N-1}}$$

By definition, each variable of a multilinear polynomial has degree 0 or 1; thus, there are N monomials and coefficients over $\log N$ variables. Let $i_0 i_1 \ldots i_{\log N-1}$ be the binary representation of i, i.e., $i = \sum_{j=0}^{\log N-1} 2^j i_j$. We use w to denote the coefficient vector, where $w[i] = w_{i_0 i_1 \ldots i_{\log N-1}}$. Similarly, define $X_i = x_0^{i_0} x_1^{i_1} \ldots x_{\log N-1}^{i_{\log N-1}}$. For $k = \sqrt{N}$, define

$$r_0 = \{X_0, X_1, \ldots, X_{k-1}\} \quad \text{and} \quad r_1 = \left\{X_{0\times k}, X_{1\times k}, \ldots, X_{(k-1)\times k}\right\}.$$

Then $X = r_0 \otimes r_1$. The polynomial evaluation is reduced to a tensor product:

$$\phi\left(x_0, x_1, \ldots, x_{\log N-1}\right) = \langle w, r_0 \otimes r_1 \rangle \tag{1}$$

Using the tensor IOP protocol of [8], one can build a polynomial commitment scheme; for completeness, we present the protocol in Protocol 1. Here, k denotes the message length of the linear code.

As shown in the protocol, to commit to a polynomial, $\mathsf{PC.Commit}$ parses the coefficient vector w as a $k \times k$ matrix W and encodes each row, yielding a $k \times n$ matrix W'. It then constructs a Merkle tree commitment for every column of W' and builds a second Merkle tree over their roots as the final commitment.

To answer a tensor query, the protocol performs two checks: a proximity check and a consistency check. The proximity check ensures that the committed matrix is close to a codeword. The consistency check ensures that $y = \langle w, r_0 \otimes r_1 \rangle$, assuming $\mathcal{C}$ is a commitment to a codeword.

Proximity check The proximity check has two steps. First, the verifier sends a random vector γ to the prover, and the prover computes the linear combination of all rows of W with coefficients γ. By linearity of the code, c_γ is a codeword. Second, the prover shows that c_γ is computed from the committed tensor codeword. To do so, the verifier randomly selects t columns, which the prover opens

together with their Merkle-tree proofs. The verifier checks that the inner product of each column with γ equals the corresponding element of c_γ. As shown in [8], if the linear code has constant relative distance, the committed matrix is close to a tensor codeword with overwhelming probability.

Consistency check The consistency check follows the same steps as the proximity check. Instead of using a random vector from the verifier, it uses the r_0 component of the tensor query $r_0 \otimes r_1$. Similarly, c_1 is a codeword with message y_1, and $\phi(x) = \langle y_1, r_1 \rangle$ by the definitions of tensor products and polynomial evaluation. As shown in [8], if the committed matrix is close to a tensor codeword, then $y = \phi(x)$ with overwhelming probability. In particular, there exists an extractor that can extract a polynomial ϕ from the commitment such that $y = \phi(x)$.

Algorithm 1 The protocol for proving twisted evaluation. [21,36,6]

Public input: The evaluation point $\mathbf{x}$, transformed into a tensor product $r_0 \otimes r_1$ in Equation 1;
Private input: The polynomial ϕ, the coefficients of ϕ are denoted by w.
Let $E_C : \mathbb{F}^k \to \mathbb{F}^n$ be the encoding function of an $[n, k, d]$ linear code. If $N = k \times k$ is not a perfect square, we pad it to the next perfect square. We use Python-style notation to select the i-th column of a matrix `mat` as $\texttt{mat}[:, i]$.

function PC.Commit(ϕ)
Parse the coefficients w as a $k \times k$ matrix W. The prover computes the encoding W' locally. In particular, W' is a $k \times n$ matrix by encoding each row of the matrix W using E_C.
for $i \in [n]$ **do**
    Compute the corresponding Merkle-tree leaf: $\mathsf{Root}_i = \mathsf{Merkle.Commit}\,(W'[:, \mathrm{i}])$.
Compute the Merkle tree root $\mathsf{Root} = \mathsf{Merkle.Commit}\,([\mathsf{Root}_0, \ldots, \mathsf{Root}_{n-1}])$ and output Root as the commitment.
function PC.Open($\phi, \mathbf{x}$)
$\mathcal{P}$ receives a random vector $\gamma \in \mathbb{F}^k$ from $\mathcal{V}$, and computes $y_\gamma = \sum_{i=0}^{k-1} \gamma[i] W[i], y_1 = \sum_{i=0}^{k-1} r_0[i] W[i]$.
$\mathcal{P}$ sends y_γ, y_1 to $\mathcal{V}$. $\mathcal{V}$ randomly samples t indices from $[n]$ as an array I and sends it to $\mathcal{P}$.
for $i \in I$ **do**
    $\mathcal{P}$ sends $W'[:, i]$ and $(\mathsf{Root}_i, \pi_i) \leftarrow \mathsf{Merkle.Open}\,([\mathsf{Root}_0, \ldots, \mathsf{Root}_{n-1}], i)$ to $\mathcal{V}$.
function PC.Verify $(\pi, \mathbf{x}, y = \phi(\mathbf{x}), \mathsf{Root})$
Compute $c_\gamma = E_C\,(y_\gamma)$ and $c_1 = E_C\,(y_1)$.
for $\forall i \in I$ **do**
    Check if $c_\gamma[i] == \langle \gamma, W'[:, i] \rangle$.
    Check if $c_1[i] == \langle r_0, W'[:, i] \rangle$.
    Check if $\mathsf{Root}_i == \mathsf{Merkle.Commit}(W'[:, i])$
    Run $\mathsf{Merkle.Verify}\,(\pi_i, \mathsf{Root}_i, \mathsf{Root})$.
Check if $y == \langle y_1, r_1 \rangle$. Output accept if all the above conditions hold; otherwise, output reject.

Theorem 1. *Protocol 1 is a polynomial commitment scheme with soundness error* $\frac{N}{\mathbb{F}} + (1 - \frac{\delta}{3})^t + (1 - \frac{2\delta}{3})^t$*, where* δ *is the relative distance of the code, and* t *is the number of opened columns in Protocol 1.*

For a linear code with constant relative distance δ, setting $t = \lambda \cdot \frac{1}{-\log_2(1-\delta/3)}$ makes the soundness error at most $2^{-\lambda}$. The proof size is $O(tk) = O(t\sqrt{N})$. By optimizing the dimensions of the coefficient matrix, the proof size can be improved to $O(\sqrt{tN})$ (see [21] for details).

Ligero [2], Brakedown [21], Orion [36], and EA [6] all adhere to the construction recipe described above, differing primarily in their choice of underlying linear code. Ligero relies on Reed–Solomon codes, Brakedown and Orion adopt the Brakedown code [21], while EA employs Expander-Accumulation codes [9]. These distinct code choices yield different trade-offs between prover time and proof size across the resulting PCS constructions.

2.6 LDPC Matrix

Definition 4. *Let* $A \in \{0,1\}^{n \times m}$ *be a binary matrix. We call* A *a low-density parity-check (LDPC) matrix [20] if the number of non-zero entries in* A *is much smaller than* $n \cdot m$*. Formally,* $\frac{\mathsf{weight}(A)}{n \cdot m} \ll 1$*, where* $\mathsf{weight}(A) = |\{(i,j) : A_{i,j} = 1\}|$ *is the number of ones in* A*.*

LDPC matrices are an important tool in coding theory and cryptography [16,13]. In this paper, we focus on *row-regular* LDPC matrices, where every row of A has Hamming weight exactly D. We use two closely related distributions over such matrices.

Definition 5 (D-row-regular LDPC ensemble). *Let* $A \in \{0,1\}^{n \times m}$ *be a random binary matrix whose* n *row supports are sampled* independently *and uniformly at random from the* $\binom{m}{D}$ *binary vectors of Hamming weight exactly* D*. We call the resulting distribution over* A *the* D-row-regular LDPC ensemble*. Note that under this ensemble two rows of* A *may coincide.*

Definition 6 (Collision-free D-row-regular LDPC ensemble). *Let* $A \in \{0,1\}^{n \times m}$ *be a random binary matrix whose* n *row supports are sampled* without replacement *and uniformly at random from the* $\binom{m}{D}$ *binary vectors of Hamming weight exactly* D*, so that no two rows of* A *are identical. We call the resulting distribution over* A *the* collision-free D-row-regular LDPC ensemble*.*

3 Technical Overview

A large class of efficient polynomial commitment schemes (PCS), including Brakedown [21], Orion [36], Ligero [2], and EA [6], follow a common paradigm: they build commitments and proofs from linear error-correcting codes combined with tensor IOPs. In these schemes, the dominant prover cost arises from repeatedly encoding rows of large witness matrices. Consequently, the encoding efficiency of the underlying code largely determines the overall prover time of the PCS.

High-level idea Lightning accelerates error-correcting-code encoding by applying the base code to only a *compressed linear sketch* of the message, rather than to the full message. Concretely, we first apply an efficiently computable linear map that reduces the message length from n to αn for some $\alpha < 1$. We then encode this compressed representation using the base code and concatenate it with the original message in a systematic manner. This design reduces the computational cost of the base-encoding step by a factor of α, replacing it with a lightweight linear map while still preserving linear distance guarantees.

From a base code to the Lightning code Let $\mathcal{C}_D$ be a linear code over $\mathbb{F}_q$ with rate ρ and relative distance δ. Given a message $x \in \mathbb{F}_q^n$, Lightning applies a linear mapping $A : \mathbb{F}_q^n \to \mathbb{F}_q^{\alpha n}$ and defines the Lightning codeword as

$$\mathcal{C}_L(x) := x \,\|\, \mathcal{C}_D(xA),$$

where "$\|$" denotes concatenation. As long as the linear transform A is inexpensive to compute, encoding $\mathcal{C}_L(x)$ is substantially cheaper than encoding $\mathcal{C}_D(x)$, since the base code is applied only to a message of length αn.

Sparsity preservation The key requirement on A is that it preserves nontrivial information about vectors with relative weight up to a prescribed fraction: informally, A should map every nonzero x with support at most βn to $xA \neq 0$ (Definition 7 formalizes this as *β-sparsity preservation*). This property is introduced by efficiently encodable codes: the Spielman code [34] and Brakedown [21] use matrices that ensure a sufficiently low-weight nonzero vector remains nonzero after matrix multiplication. Lightning leverages this property differently, using such a matrix as a compressed linear sketch before encoding with an arbitrary large-distance base code. As we show in Lemma 1, this yields a generic code construction with constant relative distance, with the achievable distance improving as the certified β grows.

Designing the linear transform To ensure that xA can be computed efficiently, a natural choice is to take A as a sparse binary matrix—specifically, a Low-Density Parity-Check (LDPC) matrix [20]. We show that if $A \in \{0,1\}^{n \times \alpha n}$ is sampled uniformly from the D-row-regular LDPC ensemble, then, with high probability, it is β-sparsity-preserving for an explicit $\beta = f(\alpha, D)$.

The unique column property Rather than analyzing sparsity preservation directly, Sections 4.2–4.4 work with a strictly stronger, purely combinatorial condition on A that is easier to reason about: the *unique column property*, which asks that every sufficiently small row subset S have some column with exactly one nonzero entry among the rows in S. Such a column immediately certifies $xA \neq 0$ for any x supported on S, so establishing this property for all small S is enough to conclude sparsity preservation.

Intuition: a balls-into-buckets view The unique-column property has a simple balls-into-buckets interpretation. Each row in a set S throws D balls into D distinct columns (buckets) chosen uniformly at random. When $D|S|$ is well below the number of buckets αn, most balls land alone, so a bucket containing exactly one ball is likely to exist. Such a bucket is precisely a unique column for S.

Analysis roadmap Section 4.2 formalizes this viewpoint. Section 4.3 then adapts Silver's classical double-union-bound analysis [16] to obtain a rigorous but initially loose baseline. Section 4.4 tightens this baseline through a counting-based analysis that retains the full occupancy profile rather than relaxing to covered-column sets. Finally, Section 4.5 refines the bound for collision-free LDPC sampling.

Efficiency Comparison Having established that the Lightning code preserves constant relative distance, we now turn to its encoding cost. Since each row of A contains exactly D nonzero entries (all equal to 1), computing xA requires only Dn field additions and no field multiplications. In practice, D can be as small as 7 or 8. By contrast, the fastest Brakedown parameters we searched require about $8.07n$ field multiplications and $8.07n$ field additions. Moreover, the encoding cost of the base code $\mathcal{C}_D$ is reduced by a factor of α. As a result, the overall encoding of the Lightning code is substantially faster than that of its base code.

Our experimental results confirm that Lightning PCS substantially reduces commitment time relative to existing coding-based PCS constructions. This improvement is primarily driven by Lightning's reduced reliance on expensive field multiplications, compared with prior efficiently encodable linear codes such as Brakedown [21], Expander-Accumulation (EA) [9,6], and Reed–Solomon codes (as used in Ligero [2]). In particular, Brakedown and EA incur many field multiplications through repeated sparse matrix-vector multiplications with random field coefficients, while Reed–Solomon encoding relies on NTTs and likewise involves extensive field multiplications.

4 Methodology

We first define the Lightning encoding procedure as a systematic code that appends a base-code encoding of a compressed linear sketch (Section 4.1). In Section 4.2, we prove that Lightning inherits constant relative distance provided that A satisfies the *unique-column* property. Section 4.3 then adapts Silver's classical double-union-bound argument to our binary row-regular LDPC ensemble. Its asymptotic guarantee is useful as a baseline, but its concrete β is much smaller than the construction tolerates. Section 4.4 tightens this same analysis with a direct counting argument, and Section 4.5 transfers the count to our collision-free ensemble. Finally, in Section 4.6, we combine the resulting bound with an explicit encoding-cost analysis to select concrete parameters (α, D) that minimize encoding cost while meeting a target relative distance and target code-failure probability.

4.1 Lightning Encoding

Lightning code accelerates ECC encoding by applying the base code on only a *compressed linear sketch* of the message, rather than the full message itself. Concretely, we first apply an efficiently computable linear map that reduces the message dimension to αn for some $\alpha < 1$. We then encode this compressed representation using the base code and concatenate it with the original message in a systematic manner.

Let $\mathcal{C}_D$ be a linear code over $\mathbb{F}_q$ with rate ρ and relative distance δ. Given a message $x \in \mathbb{F}_q^n$, Lightning applies a linear mapping $A : \mathbb{F}_q^n \to \mathbb{F}_q^{\alpha n}$ and defines the Lightning codeword as

$$\mathcal{C}_L(x) := x \,\|\, \mathcal{C}_D(xA),$$

where "$\|$" denotes concatenation. As long as the linear transform A is inexpensive to compute, encoding $\mathcal{C}_L(x)$ is substantially cheaper than encoding $\mathcal{C}_D(x)$, since the base code is applied only to a message of length αn.

Sparsity-preserving sketches. We observe that efficiently encodable codes such as the Spielman code [34] and Brakedown [21] use matrices with a useful property: a nonzero vector whose relative weight is at most a prescribed fraction remains nonzero after multiplication by the matrix. We call this property *sparsity preservation*. We observe that a sparsity-preserving matrix is well suited for constructing a compressed linear sketch. Given a message x, Lightning first computes xA and then encodes xA with a base code of large relative distance. Since xA has length αn, the expensive base-code encoder operates only on a compressed message, accelerating encoding at the cost of relative distance. Lightning retains the original message in the codeword, while sparsity preservation ensures that a nonzero message in this support range remains visible in the sketch. To make the sketch fast to compute, we instantiate A from a binary $\{0, 1\}$ row-regular LDPC ensemble. Every selected coefficient of A is then one, so computing xA takes Dn field additions and no field multiplications.

We formalize this requirement as follows.

Definition 7 (Sparsity Preservation). *Matrix $A \in \mathbb{F}_q^{n \times \alpha n}$ is* β-sparsity-preserving *if for every nonzero vector $x \in \mathbb{F}_q^n$ with $0 < \|x\|_0 < \beta n$, it holds that $xA \neq 0$.*

The following lemma shows that if A satisfies sparsity preservation, then the resulting Lightning code $\mathcal{C}_L$ achieves constant relative distance.

Lemma 1. *Let $A \in \mathbb{F}_q^{n \times \alpha n}$ be β-sparsity-preserving (Definition 7). Let $\mathcal{C}_D$ be any error-correcting code with rate ρ and relative distance δ. Define the Lightning code $\mathcal{C}_L : \mathbb{F}_q^n \to \mathbb{F}_q^{n+\alpha n/\rho}$ by*

$$\mathcal{C}_L(x) = x \,\|\, \mathcal{C}_D(xA)$$

Then for every $x \in \mathbb{F}_q^n \setminus \{0\}$,

$$\|\mathcal{C}_L(x)\|_0 \;\geq\; \min\left\{\beta,\; \frac{\alpha\delta}{\rho}\right\} \cdot n$$

Consequently, $\mathcal{C}_L$ *has relative distance*

$$\mathsf{rel_dist}(\mathcal{C}_L) \;\geq\; \min\left\{\frac{\beta\rho}{\rho+\alpha}, \frac{\delta\alpha}{\rho+\alpha}\right\}$$

Proof. Since $\mathcal{C}_D$ has rate ρ, the codeword $\mathcal{C}_D(xA)$ has length $\alpha n/\rho$. We consider two cases.

Case 1: $\|x\|_0 < \beta n$. By β-sparsity preservation, we have $xA \neq 0$, and thus

$$\|\mathcal{C}_D(xA)\|_0 \;\geq\; \delta \cdot \frac{\alpha n}{\rho}$$

Therefore,

$$\|\mathcal{C}_L(x)\|_0 \;\geq\; \|\mathcal{C}_D(xA)\|_0 \;\geq\; \frac{\alpha\delta}{\rho} \cdot n$$

Case 2: $\|x\|_0 \geq \beta n$. Then

$$\|\mathcal{C}_L(x)\|_0 \;\geq\; \|x\|_0 \;\geq\; \beta n$$

Combining the two cases yields

$$\|\mathcal{C}_L(x)\|_0 \;\geq\; \min\left\{\beta,\; \frac{\alpha\delta}{\rho}\right\} \cdot n$$

Finally, since $\mathcal{C}_L$ has block length $n + \alpha n/\rho = n(\rho+\alpha)/\rho$, we obtain

$$\mathsf{rel_dist}(\mathcal{C}_L) = \min_{x\neq 0} \frac{\|\mathcal{C}_L(x)\|_0}{n + \alpha n/\rho} \;\geq\; \min\left\{\frac{\beta\rho}{\rho+\alpha}, \frac{\delta\alpha}{\rho+\alpha}\right\}$$

We now turn to constructing a matrix A that is β-sparsity-preserving while still allowing xA to be computed efficiently. As mentioned in Section 3, we instantiate A as a uniformly random D-row-regular binary LDPC matrix. Note that under this choice, computing xA requires only $O(nD)$ field additions.

The remaining task is to show that such a random matrix $A \in \{0,1\}^{n\times\alpha n}$ satisfies the β-sparsity-preserving property with high probability, for an explicit parameter $\beta = f(\alpha, D)$.

4.2 Unique Column Property

To simplify the analysis, we prove a strictly stronger combinatorial condition, which we call the *unique column property*: for every row subset $S \subseteq [n]$ with $|S| \leq \beta n$, there exists a column of A that has exactly one nonzero entry among the rows in S.

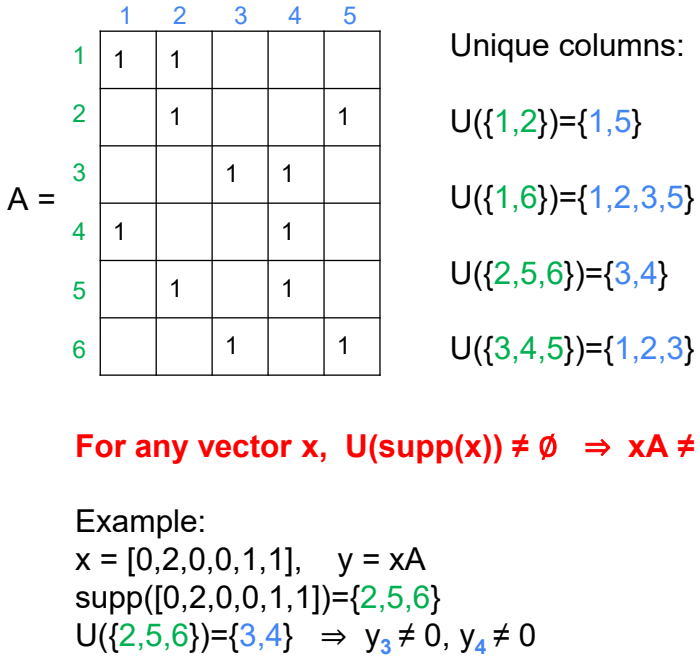


Fig. 1. For a 6×5 2-row-regular LDPC matrix A, we illustrate several instances of $U_A(S)$ for different choices of S. The existence of a unique column for set S guarantees that $xA \neq 0$ for any nonzero vector x with $\mathsf{supp}(x) = S$.

The key intuition is that the product xA can be viewed as a linear combination of the rows of A, with coefficients given by the entries of x. Let $S = \mathsf{supp}(x)$. Under the unique column property, S admits a *unique column* c, meaning that exactly one row $r^\star \in S$ has a nonzero entry in column c. Consequently, the c-th coordinate of xA equals $x_{r^\star} \neq 0$, implying $xA \neq 0$ and hence establishing β-sparsity preservation.

Formally, for a row set $S \subseteq [n]$, we define the column weight

$$\mathrm{wt}_S(c) := \sum_{r \in S} A_{r,c}.$$

A column c is called *unique* for S if $\mathrm{wt}_S(c) = 1$, and we denote the set of such columns by

$$U_A(S) := \{c \in [\alpha n] : \mathrm{wt}_S(c) = 1\}.$$

We have the following lemma:

Lemma 2 (Unique column $\Rightarrow$ β-sparsity-preserving). *If for every $S \subseteq [n]$ with $|S| \leq \beta n$ we have $U_A(S) \neq \emptyset$, then for every prime q, every nonzero $x \in \mathbb{F}_q^n$ with $\|x\|_0 < \beta n$ we have $xA \neq 0$.*

Proof. Let $S = \mathsf{supp}(x)$. Pick $c \in U_A(S)$ and let $r^\star \in S$ be the unique row with $A_{r^\star,c} = 1$. Then $(xA)_c = \sum_{r \in S} x_r A_{r,c} = x_{r^\star} \neq 0$, hence $xA \neq 0$.

Lemma 2 shows that the unique column property implies β-sparsity preservation. Therefore, it remains to show that $U_A(S) \neq \emptyset$ for all row subsets S with $|S| \leq \beta n$ with high probability.

We provide some intuition for why, for a uniformly random D-row-regular matrix $A \in \{0,1\}^{n \times \alpha n}$, a small row subset S is very likely to satisfy $U_A(S) \neq \emptyset$. View the αn columns as αn buckets. Each row in S places D balls (ones) into D *distinct* buckets chosen uniformly at random, so the total number of balls is $D|S|$.

When $D|S| \ll \alpha n$, the process operates in a sparse regime where collisions are rare: most balls land in previously empty buckets. Consequently, the number of occupied buckets is close to $D|S|$, making it unlikely that every occupied bucket has load at least 2. Therefore, with overwhelming probability, there exists at least one bucket with load exactly 1, corresponding to a unique column in $U_A(S)$.

4.3 Double-Union-Bound Failure Analysis

The unique-column failure event for random row-regular LDPC matrices has already been studied in the LDPC literature. In particular, Appendix C, Theorem 7 of Silver [16] shows that a random row-regular sparse binary matrix achieves the required unique-column neighbor property with high probability. We adapt its union-bound argument to our binary D-row-regular ensemble. Its two decisive union bounds are over all row sets S of size s, and, for each fixed S, over all covered-column sets T of size $Ds/2$. The latter follows after replacing $U_A(S) = \emptyset$ by the larger event that every one of the Ds incidences lies in T; the former then ranges over the $\binom{n}{s}$ possible sets S. The analysis of [23] uses the same double-union-bound template. This template is convenient to analyze but relatively loose.

To establish the unique-column property of Section 4.2, we upper bound the failure probability

$$\Pr[\exists\, S \subseteq [n],\ |S| \leq \beta n,\ U_A(S) = \emptyset]\,.$$

The following lemma upper bounds the failure probability conditioned on the row subset size s.

Lemma 3. *Let $A \in \{0,1\}^{n \times \alpha n}$ be sampled from the D-row-regular LDPC ensemble of Definition 5 (independent row sampling). For any integer $s \in [1, \frac{2\alpha n}{D}]$, we have*

$$\Pr\left[\exists\, S \subseteq [n],\, |S| = s, |U_A(S)| = \emptyset\right] \ \leq\ \left(\frac{Cs}{n}\right)^{s(\frac{D}{2}-1)},$$

where $C = e^{\frac{D+2}{D-2}} \left(\frac{D}{2\alpha}\right)^{\frac{D}{D-2}}$ is a constant.

Proof. We first fix a set $S \subseteq \{1, \cdots, n\}$ with $|S| = s$. And we upper bound the probability of "bad" event $|U_A(S)| = \emptyset$.

Define covered column set of S as $H_S = \{c | c \in \{1, \cdots, \alpha n\}, \sum_{r \in S} A_{r,c} \geq 1\}$.

If $|U_A(S)| = \emptyset$ occurs, then for each column c, $\sum_{r \in S} A_{r,c} = 0$ or $\sum_{r \in S} A_{r,c} \geq 2$, hence $\sum_{r \in S} \sum_{c \in \{1, \cdots, \alpha n\}} A_{r,c} \geq 2|H_S|$.

Since each row in A has D ones, $\sum_{r \in S, c \in \{1, \cdots, \alpha n\}} A_{r,c} = Ds$; together with the preceding inequality, this shows that $|U_A(S)| = \emptyset$ entails $|H_S| \leq Ds/2$. We next upper bound the probability of $|U_A(S)| = \emptyset$ by upper bounding the probability of $|H_S| \leq Ds/2$.

If $|H_S| \leq \frac{Ds}{2}$, then there exists a set $T \subseteq \{1, \ldots, \alpha n\}$ of size $\frac{Ds}{2}$ such that $H_S \subseteq T$. For each row index r, define its set of nonzero columns as

$$\mathsf{nz}(r) = \{c \in \{1, \ldots, \alpha n\} : A_{r,c} = 1\}.$$

Therefore, we have the following implication:

$$\begin{aligned} |U_A(S)| = \emptyset &\implies |H_S| \leq \frac{Ds}{2} \\ &\implies \exists\, T \subseteq \{1, \ldots, \alpha n\},\ |T| = \frac{Ds}{2} \text{ such that } \forall r \in S,\ \mathsf{nz}(r) \subseteq T. \end{aligned}$$

Fix an arbitrary set $T \subseteq \{1, \ldots, \alpha n\}$ with $|T| = \frac{Ds}{2}$. For a given r, the probability that $\mathsf{nz}(r) \subseteq T$ is

$$\Pr[\mathsf{nz}(r) \subseteq T] = \frac{\binom{\frac{Ds}{2}}{D}}{\binom{\alpha n}{D}} = \prod_{i=0}^{D-1} \frac{\frac{Ds}{2} - i}{\alpha n - i} \leq \left(\frac{\frac{Ds}{2}}{\alpha n}\right)^D.$$

Taking a union bound over all T with $|T| = Ds/2$ and using independence across $r \in S$ gives

$$\begin{aligned} \Pr[U_A(S) = \emptyset] &\leq \sum_{\substack{T \subseteq [\alpha n] \\ |T| = Ds/2}} \Pr\left[\forall r \in S,\ \mathsf{nz}(r) \subseteq T\right] \\ &\leq \binom{\alpha n}{Ds/2} \left(\frac{Ds/2}{\alpha n}\right)^{Ds}. \end{aligned}$$

We now take a union bound over the $\binom{n}{s}$ choices of S. Applying the standard Stirling bound $\binom{u}{v} \leq (eu/v)^v$ to both resulting binomial coefficients yields

$$\Pr\left[\exists\, S \subseteq [n],\, |S| = s, |U_A(S)| = \emptyset\right] \ \leq\ \left(\frac{Cs}{n}\right)^{s(\frac{D}{2} - 1)} \tag{2}$$

where

$$C = e^{\frac{D+2}{D-2}} \left(\frac{D}{2\alpha}\right)^{\frac{D}{D-2}}.$$

Lemma 3 bounds the failure probability for a *fixed* subset size s. We now extend this bound to all subsets of size at most βn by taking a union bound over $s = 1, 2, \ldots, \lfloor \beta n \rfloor$. By choosing $\beta = \frac{0.98}{C}$, the resulting summation is dominated by a rapidly decaying geometric series, which yields an overall failure probability of $O(n^{3-D})$. This gives the following theorem.

Theorem 2 (Probability of no unique column). $\forall$ *integer* $D \geq 4$, *let* $A \in \{0,1\}^{n \times \alpha n}$ *be sampled from the* D*-row-regular LDPC ensemble of Definition 5 (independent row sampling).*

$$\Pr\left[\exists\, S \subseteq [n],\, |S| \leq \lfloor \frac{0.98n}{C} \rfloor, |U_A(S)| = \emptyset\right] \;\leq\; O(n^{3-D}),$$

where constant $C = e^{\frac{D+2}{D-2}} \left(\frac{D}{2\alpha}\right)^{\frac{D}{D-2}}$.

The proof appears in Appendix A. More importantly, Lemma 3 makes the baseline's limitation on large sets explicit. Its right-hand side is nontrivial only while $Cs/n < 1$. Hence, when s approaches βn, this argument can certify only $\beta < 1/C$: once β exceeds $1/C$, the bound for the largest set sizes is already at least one. This limitation comes from the union-bound relaxation in Lemma 3. For $s = \Theta(n)$, the union bound over covered-column-set events is particularly loose.

4.4 Counting-Based Failure Analysis

The baseline of Section 4.3 is rigorous, but its large-set weakness is structural. For a fixed S, it first replaces the event $U_A(S) = \emptyset$ by the larger event that all Ds incidences are contained in some set T of only $Ds/2$ columns. It then unions over every such T. Thus it counts many configurations that contain singleton columns and counts the same configuration through many supersets T. As observed after Theorem 2, this produces the Cs/n barrier that makes the bound vacuous once s exceeds n/C.

Our goal is a tighter upper bound on $\Pr_{\mathrm{rr}}[U_A(S) = \emptyset]$ for each fixed row set S of size s. [1] We first turn this probability into an exact counting problem in a simpler occupancy experiment.

In the row-regular experiment, the s rows independently each throw D balls into pairwise distinct bins in $[m]$, where $m := \alpha n$. We relax this row structure by instead throwing the $L := Ds$ labelled balls independently, one at a time, into $[m]$. The resulting no-singleton-bin event, denoted $\mathcal{E}_{\mathrm{ind}}(S)$, is the occupancy analogue of $U_A(S) = \emptyset$. Its probability is convenient to count exactly; the difference from the row-regular experiment is restored by an explicit correction factor.

Let $N_{m,L}$ be the number of assignments of $L = Ds$ labelled incidences to m bins in which no bin receives exactly one incidence. This is exactly the number of bad assignments in the relaxed experiment. Since each assignment has

[1] Here $\Pr_{\mathrm{rr}}$ is over the row-regular ensemble in which the rows independently choose uniform weight-D supports. We use $\Pr_{\mathrm{ind}}$ below for the independent-incidence relaxation; its no-singleton event is denoted separately by $\mathcal{E}_{\mathrm{ind}}(S)$.

probability m^{-Ds}, that experiment satisfies

$$\Pr_{\text{ind}}[\mathcal{E}_{\text{ind}}(S)] = \frac{N_{m,Ds}}{m^{Ds}}. \tag{3}$$

We now calculate $N_{m,L}$ with a generating function. For one labelled bin, the exponential generating function of allowed occupancies is

$$1 + \sum_{\ell \geq 2} \frac{z^\ell}{\ell!} = e^z - z.$$

The product $(e^z - z)^m$ independently chooses an allowed occupancy for each of the m labelled bins, and multiplication by $L!$ assigns the L labelled incidences to these occupancies. Thus

$$N_{m,L} := L!\,[z^L](e^z - z)^m, \tag{4}$$

where $[z^L]f(z)$ denotes the coefficient of z^L in $f(z)$.

To calculate $N_{m,L}$, we use the standard associated-Stirling identity [14]:

$$\frac{(e^z - z - 1)^k}{k!} = \sum_{L \geq 0} S_{\geq 2}(L,k)\frac{z^L}{L!}.$$

Here $S_{\geq 2}(L,k)$ denotes the associated Stirling number of the second kind [14]. The binomial theorem and the displayed identity give

$$\begin{aligned}(e^z - z)^m &= \sum_{k=0}^{m} \binom{m}{k} (e^z - z - 1)^k \\ &= \sum_{k=0}^{m} \binom{m}{k} k! \sum_{\ell \geq 0} S_{\geq 2}(\ell,k)\frac{z^\ell}{\ell!} \\ &= \sum_{\ell \geq 0} \left(\sum_{k=0}^{\lfloor \ell/2 \rfloor} (m)_k S_{\geq 2}(\ell,k) \right) \frac{z^\ell}{\ell!}.\end{aligned}$$

The second equality substitutes the associated-Stirling identity, and the third uses $\binom{m}{k}k! = (m)_k$, where $(m)_k := m(m-1)\cdots(m-k+1)$ is the falling factorial.

Applying $L![z^L]$ to both sides, as in Equation (4), yields

$$N_{m,L} = \sum_{k=0}^{\lfloor L/2 \rfloor} (m)_k S_{\geq 2}(L,k). \tag{5}$$

To evaluate $N_{m,L}$ via Equation (5), we compute the associated Stirling numbers of the second kind from their recursive definition:

$$\begin{aligned}S_{\geq 2}(L,k) &= kS_{\geq 2}(L-1,k) + (L-1)S_{\geq 2}(L-2,k-1), \\ S_{\geq 2}(0,0) &= 1.\end{aligned} \tag{6}$$

We next relate the probabilities in the independent-incidence and row-regular experiments. Fix S and partition the Ds independent choices into its s ordered row blocks of size D. Let $\mathcal{R}_{\text{distinct}}(S)$ be the event that, within every block, the D chosen columns are pairwise distinct. Conditioning the relaxed experiment on $\mathcal{R}_{\text{distinct}}(S)$ gives the row-regular experiment. Therefore

$$\Pr_{\text{rr}}[U_A(S) = \emptyset] = \Pr_{\text{ind}}[\mathcal{E}_{\text{ind}}(S) \mid \mathcal{R}_{\text{distinct}}(S)] \leq \frac{\Pr_{\text{ind}}[\mathcal{E}_{\text{ind}}(S)]}{\Pr_{\text{ind}}[\mathcal{R}_{\text{distinct}}(S)]}.$$

It remains to bound the denominator. Since the blocks are independent,

$$\Pr_{\text{ind}}[\mathcal{R}_{\text{distinct}}(S)] = p_D^s, \qquad p_D := \frac{(m)_D}{m^D}.$$

Under $m = \alpha n$, $n \geq 2(D-1)/\alpha$, and $s \leq \beta n$,

$$-\log p_D \leq \sum_{i=0}^{D-1} \frac{i}{m-i} \leq \frac{D(D-1)}{2(m-D+1)} \leq \frac{D(D-1)}{\alpha n}.$$

Therefore

$$p_D^{-s} \leq \exp\left(\frac{D(D-1)\beta}{\alpha}\right).$$

Substituting this expression for the conditioning probability and applying Equation (3) gives

$$\begin{aligned}\Pr_{\text{rr}}[U_A(S) = \emptyset] &\leq p_D^{-s} \Pr_{\text{ind}}[\mathcal{E}_{\text{ind}}(S)] \\ &\leq \exp\left(\frac{D(D-1)\beta}{\alpha}\right) \frac{N_{m,Ds}}{m^{Ds}}.\end{aligned} \tag{7}$$

Consequently, defining

$$H_s := \binom{n}{s} \frac{N_{m,Ds}}{m^{Ds}},$$

the outer union bound required by Section 4.2 gives

$$\Pr_{\text{rr}}[\exists\, S \subseteq [n] : 2 \leq |S| \leq b,\ U_A(S) = \emptyset] \leq \exp\left(\frac{D(D-1)\beta}{\alpha}\right) \sum_{s=2}^{b} H_s. \tag{8}$$

The bound in (7) eliminates the earlier loose union bound over covered sets T in the analysis of Section 4.3.

4.5 Collision-Free Sampling

We further observe that the $s = 2$ term H_2 in Equation (8) can dominate the failure bound for some parameter choices. This event occurs only when two rows have the same support. We eliminate its contribution by instead sampling A

from the collision-free D-row-regular LDPC ensemble of Definition 6. [2] This sampling rule rules out failures of size $s = 2$, so the probability sum starts at $s = 3$. We now quantify its remaining cross-row correction and combine it with the fixed-set bound of Equation (7), thereby obtaining Theorem 3.

Theorem 3 (Direct counting bound). *For $D \geq 4$ and every field $\mathbb{F}$, let $A \in \{0,1\}^{n\times m}$ be sampled uniformly without replacement from the $M = \binom{m}{D}$ weight-D row types. If $m = \alpha n$, $n \geq 2(D-1)/\alpha$, and $b = \lfloor \beta n \rfloor$ satisfies $b(b-1) < 2M$, define, for each $3 \leq s \leq b$,*

$$G_s := \binom{n}{s} \frac{1}{1 - \frac{s(s-1)}{2M}} \exp\left(\frac{D(D-1)\beta}{\alpha}\right) \frac{1}{m^{Ds}} \sum_{k=0}^{\lfloor Ds/2 \rfloor} (m)_k S_{\geq 2}(Ds, k). \tag{9}$$

Then

$$\Pr_{\mathrm{cf}}\left[\exists\, x \in \mathbb{F}^n \setminus \{0\} : \|x\|_0 \leq b,\ xA = 0\right] \leq \sum_{s=3}^{b} G_s. \tag{10}$$

Proof. The unique-column property implies the desired kernel statement by Lemma 2. Fix $3 \leq s \leq b$ and a set S of s rows. Under $\Pr_{\mathrm{rr}}$, the supports of these rows are independent and uniform over the M weight-D row types. Let $\mathcal{R}_{\text{no-duplicates}}(S)$ be the event that they are all distinct. Its probability is

$$g_s := \frac{(M)_s}{M^s} = \prod_{i=0}^{s-1}\left(1 - \frac{i}{M}\right) \geq 1 - \frac{s(s-1)}{2M},$$

where the final inequality uses $\prod_i (1 - x_i) \geq 1 - \sum_i x_i$. The collision-free distribution is the row-regular distribution conditioned on $\mathcal{R}_{\text{no-duplicates}}(S)$. Therefore

$$\begin{aligned}\Pr_{\mathrm{cf}}[U_A(S) = \emptyset] &\leq g_s^{-1} \Pr_{\mathrm{rr}}[U_A(S) = \emptyset] \\ &\leq \frac{1}{1 - \frac{s(s-1)}{2\binom{m}{D}}} \exp\left(\frac{D(D-1)\beta}{\alpha}\right) \frac{N_{m,Ds}}{m^{Ds}}.\end{aligned}$$

Finally, union bound over the $\binom{n}{s}$ choices of S and then sum over $3 \leq s \leq b$. By the definition of G_s, the resulting upper bound is $\sum_{s=3}^{b} G_s$, as claimed.

4.6 Cost Analysis and Concrete Parameters

Given n, α, D, β, and a base-code operating point (ρ, δ), Theorem 3 computes the code-failure upper bound

$$P_{\mathsf{fail}}(n, \alpha, D, \beta) := \sum_{s=3}^{\lfloor \beta n \rfloor} G_s.$$

[2] We write $\Pr_{\mathrm{cf}}$ for probabilities in this collision-free ensemble, in contrast with $\Pr_{\mathrm{rr}}$ and $\Pr_{\mathrm{ind}}$ from Section 4.4.

Whenever $P_{\mathsf{fail}}(n, \alpha, D, \beta) \le \mathcal{E}$, the resulting Lightning code has, except with probability at most $\mathcal{E}$, relative distance at least

$$\mathsf{rel_dist}(\mathcal{C}_L) \ge \min\left\{ \frac{\beta\rho}{\rho+\alpha}, \frac{\delta\alpha}{\rho+\alpha} \right\}.$$

For a base-code operating point (ρ, δ) with per-symbol cost $\mathsf{cost}(\rho, \delta)$, Lightning first computes the binary LDPC sketch xA at a cost of $Dn\,\mathbb{F}_+$, then encodes its length-αn output with $\mathcal{C}_D$. Thus the total encoding cost on a length-n message is $\mathsf{cost}(\rho, \delta)\alpha n\,\mathbb{F}_\times$ and $(\mathsf{cost}(\rho, \delta)\alpha + D)n\,\mathbb{F}_+$.

Given a target relative distance and a fixed tuple $(\rho, \delta, \alpha, D)$, we choose the largest β for which the bound of Theorem 3 is at most the target failure probability. We find this value by numerical bisection over β. For fixed α, β, D, every summand in Equation (10) is (weakly) decreasing in n, so evaluating the bound at the selected threshold value of n also certifies every larger message length.

For each target relative distance, we range over the Brakedown base-code operating points of Haböck et al. [23] in Table 2 and the Lightning parameters (α, D), then retain the feasible row with minimum equivalent encoding cost. Table 3 gives the resulting parameters.

Table 3. Cost-optimal Lightning parameters from Theorem 3 at $n = 2^{20}$ and failure target 2^{-100}.

Rate	Distance	Encoding Cost $\mathbb{F}_\times$	Encoding Cost $\mathbb{F}_+$	Pr[failure]	α	D	β	ρ	δ
0.881	0.005	$1.13n$	$10.13n$	$2^{-100.755}$	0.082497	9	0.005676	0.610	0.05
0.795	0.010	$2.16n$	$10.16n$	$2^{-100.862}$	0.157691	8	0.012585	0.610	0.05
0.733	0.015	$3.56n$	$11.56n$	$2^{-101.386}$	0.225947	8	0.020475	0.619	0.06
0.667	0.020	$4.88n$	$12.88n$	$2^{-113.070}$	0.309502	8	0.030000	0.619	0.06
0.581	0.025	$7.04n$	$14.04n$	$2^{-100.001}$	0.446572	7	0.043036	0.619	0.06
0.560	0.030	$8.33n$	$16.33n$	$2^{-100.096}$	0.456202	8	0.053556	0.581	0.07

Table 4 compares this search with the Section 4.3 baseline.

Table 4. Concrete parameter search at $n = 2^{20}$ and failure probability 2^{-100}. We compare the union-bound analysis of Section 4.3 with the counting-based bound of Theorem 3. The Section 4.3 search has no feasible parameter set for any target relative distance at least 0.011.

Target distance	Section 4.3			Theorem 3		
			Equivalent			Equivalent
	α	D	encoding cost	α	D	encoding cost
0.005	0.3429	9	$6.49n$	0.0825	9	$3.66n$
0.010	0.7653	9	$11.72n$	0.1577	8	$4.69n$

Table 4 shows that the direct counting bound of Theorem 3 is substantially tighter than the classical double-union-bound analysis of Section 4.3 at concrete

parameters. At target distance 0.005, it reduces the minimum equivalent encoding cost from $6.49n$ to $3.66n$ ($1.77\times$); at target distance 0.010, it reduces the cost from $11.72n$ to $4.69n$ ($2.50\times$). Moreover, the double-union-bound analysis has no feasible parameter set for target distances at least 0.011, whereas the direct count certifies the parameters in Table 3 through target distance 0.030. The tighter certificate permits a smaller sketch ratio α at a fixed target distance, which directly lowers the encoding cost.

4.7 Adding Zero-Knowledge

To make our PCS zero-knowledge, we apply the masking technique of [7,36]. For each $i \in [k]$, the prover samples $R_i \leftarrow \mathbb{F}^n$ uniformly at random and replaces the $i-$th row's codeword $E_C(W[i])$ with the randomized encoding $\big(E_C(W[i]) + R_i \,\big\|\, R_i\big)$ of length $2n$, where $\|$ denotes concatenation. By construction, for any column index $j \in [n]$, the value at position j is a uniformly random field element as long as the value at position $j+n$ remains hidden (and vice versa). Accordingly, the verifier samples an opening set $\hat{I} \subset [2n]$ with $|\hat{I}| = 2t$, consisting of t indices from $[n]$ and t indices from $[n+1, 2n]$, such that for every $i \in [n]$, $\hat{I}$ contains at most one of i and $i+n$.

Viewed as a map on the extended message $(W[i], R_i) \in \mathbb{F}^k \times \mathbb{F}^n$, row i's randomized encoding $\hat{E}(W[i], R_i) := (E_C(W[i]) + R_i,\, R_i)$ is $\mathbb{F}$-linear in $(W[i], R_i)$, and a linear combination of randomized rows is again of this form with the corresponding combination of the R_i's: $\sum_i \gamma_i \hat{E}(W[i], R_i) = \hat{E}\big(\sum_i \gamma_i W[i], \sum_i \gamma_i R_i\big)$. This is what makes the column-consistency check of the proximity and consistency steps mask-invariant: for any opened index $j \le n$, both $c_\gamma[j]$ and $\langle \gamma, W'[:, j] \rangle$ pick up the same term $\sum_i \gamma_i R_i[j]$, which cancels in the equality check, so the check verifies exactly the same linear-combination relation as in the unmasked protocol. Note, however, that this cancellation argument is about the *consistency* check only; it does *not* imply that $\hat{E}$ has relative distance comparable to E_C – treated as a code of block length $2n$ over the full message space $\mathbb{F}^{k+n}$, $\hat{E}$ in fact has minimum distance as low as 1 (e.g. $W[i] = 0$ and R_i any weight-one vector), since the prover chooses R_i and could otherwise use it to disguise an invalid row as consistent on the queried columns. Soundness against this is restored precisely by the disjoint-opening constraint above: because $\hat{I}$ never contains both j and $j+n$ for the same original index, the verifier never sees enough of any single row's mask to exploit this low-distance direction, and the resulting proximity/consistency guarantee is the one established for this exact masking-with-disjoint-openings construction in [7,36], rather than a naive transfer of the unmasked distance-based soundness argument for Protocol 1.

5 Experiments

In this section, we present a comprehensive evaluation of our Lightning PCS.

Software. Following [21,6], we use a 254-bit prime in our experiments. We implement our PCS in C++, using the mcl library[3] for field operations.
Hardware. All experiments are conducted on a cloud server equipped with 90 GB of memory and 16 Intel CPU cores. We use a single thread throughout.
Baselines. In this paper, we compare the performance of our PCS against existing coding-based PCS, including Brakedown [21] (CRYPTO'23), Orion [36] (CRYPTO'22), Ligero [2] (CCS'17), and EA [6] (CRYPTO'24). For Brakedown, Orion, and EA, we adopt their official implementations. For Ligero, we rely on the implementation included in the Brakedown codebase [21].
Hyperparameters. The default Lightning configuration has relative distance $\delta = 0.005$, close to the Brakedown-fastest configuration with $\delta = 0.0043$. Using the Brakedown parameters of Haböck et al. [23] as base codes, we find no feasible Lightning configuration with target relative distance $\delta \geq 0.040$. For Brakedown, we consider Brakedown-fastest, which we obtain by a search based on the analysis of Haböck et al. [23], and their default code with $\delta = 0.020$; all other baselines use their default parameters. For Ligero, following the comparison in Brakedown [21], we use a Reed–Solomon code of rate 1/2. Following Brakedown [21] and Orion [36], all commitment schemes target 128-bit computational soundness, and the code-failure target is 2^{-100}.

5.1 Performance for Lightning PCS

Figure 2 presents the updated comparison. Orion [36] exhausted available memory when processing polynomials of degree 2^{30}.

As shown in Figure 2, for sufficiently large polynomials, Lightning's commitment time is roughly 1.8–1.9× faster than Brakedown-fastest, 2.5–3.0× faster than Orion, and 14–15× faster than EA-PC. In all existing coding-based PCS, the prover's dominant cost is committing to the polynomial, while opening is substantially faster.

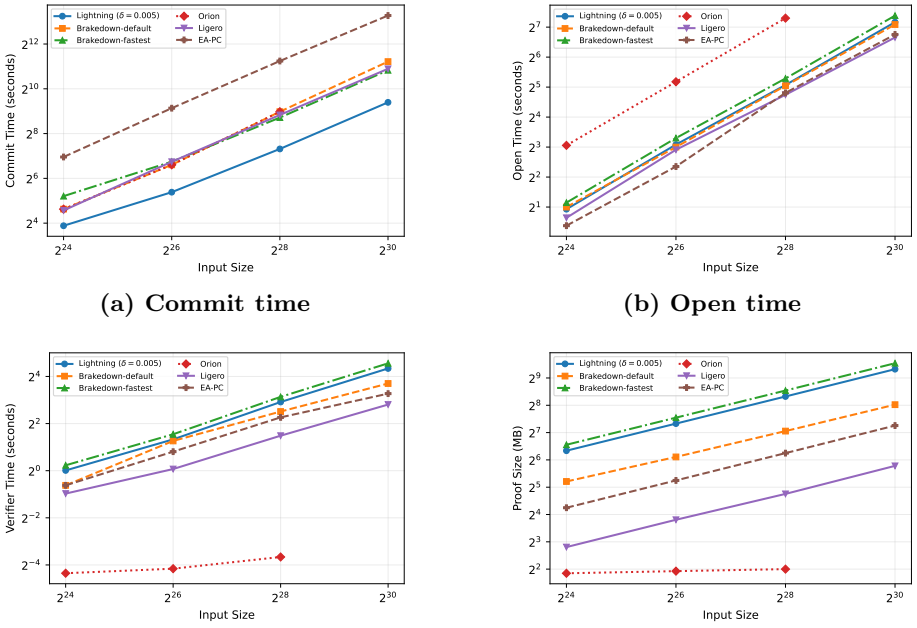
In terms of communication, Brakedown-fastest has relative distance 0.0043, so its proofs are about 1.16× larger than Lightning's at distance 0.005. All baselines except Orion have proof sizes of $O(\sqrt{N})$, whereas Orion achieves the best asymptotics with $O(\log^2 N)$ proof size and also the smallest concrete proof size.

In summary, Lightning offers a prover that is both concretely and asymptotically fast. Under the updated parameters, it also has smaller proofs and faster verification than Brakedown-fastest, whose lower relative distance requires more column openings. Orion and Ligero still achieve smaller proofs in some settings, but neither is field-agnostic, unlike Lightning.

5.2 Commit Time–Proof Size Tradeoff Across Code Distances

As shown in Table 3, different Lightning code configurations achieve different tradeoffs between encoding cost and relative distance. These choices yield Lightning PCS configurations with different commitment-time–proof-size tradeoffs.

[3] https://github.com/herumi/mcl



(a) Commit time **(b) Open time**

(c) Verifier time **(d) Proof size**

Fig. 2. Benchmark results under the updated parameters. Lightning uses relative distance 0.005, and Brakedown-fastest uses the new fastest point.

To explore the resulting prover time–proof size tradeoff, we consider four representative Lightning configurations at relative distances 0.005, 0.010, 0.020, and 0.030 in Table 3, and compare them against Brakedown-fastest at relative distance 0.0043 and Brakedown operating points at relative distances 0.020, 0.030, and 0.040. The comparison is anchored at the prior multilinear-polynomial benchmark of size 2^{28}.

The left panel of Figure 3 shows equivalent encoding costs, which are lower for Lightning throughout its feasible displayed range. The right panel translates this cost gap into the PCS proof-size–commitment-time tradeoff: its horizontal axis is proof size and its vertical axis is commitment time. The two polylines connect the Lightning and Brakedown configurations, respectively. As Lightning's relative distance increases from 0.005 to 0.030, proof size decreases from 320MB to about 53MB, while commitment time increases from about 160s to 346s. Brakedown-fastest requires about 284s to commit and, at relative distance 0.0043, has a normalized proof size of about 372MB. The Brakedown points at distances 0.020, 0.030, and 0.040 require about 349s, 410s, and 451s to commit. Their proof sizes are 80MB, about 53MB, and 40MB; the first two exactly match the Lightning configurations at those distances. The $\delta = 0.040$ endpoint has no Lightning counterpart because the parameter search finds no feasible Lightning configuration at that distance.

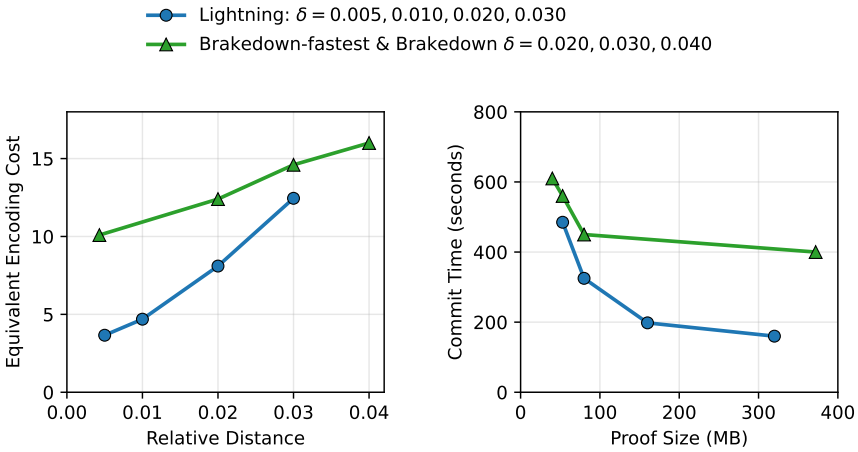


Fig. 3. Comparison of Lightning and Brakedown. Left: relative distance (x-axis) and equivalent encoding cost (y-axis). Right: proof size (x-axis) and commitment time (y-axis). Lightning uses target relative distances 0.005, 0.010, 0.020, and 0.030. Brakedown uses the fastest point from the grid search of Haböck et al. [23], followed by their 0.020, 0.030, and 0.040 parameter points. With the available Brakedown base codes of relative distance at most 0.070, Lightning has no feasible target-0.040 configuration. Points at the same relative distance have the same proof size.

This tradeoff follows directly from encoding cost and relative distance. At relative distance 0.005, Lightning requires $3.66n$ equivalent field multiplications, compared to $10.09n$ for Brakedown-fastest, a $2.8\times$ reduction. The shared normalization makes the proof-size effect of relative distance explicit.

6 Discussion

Proof size and verifier cost. Lightning targets the encoding bottleneck in the standard tensor-IOP PCS framework. Accordingly, our present PCS instantiation retains the $O(\sqrt{N})$ proof size and verifier time of the Brakedown interface. When one chooses a Lightning operating point with lower relative distance than its comparison point, it requires more openings, so proof size and verifier cost can be worse.

This tradeoff is orthogonal to the Lightning code construction. Orion's recursive technique [36] reduces the proof size and verifier time of the Brakedown-based PCS interface to $O(\log^2 N)$. Since Lightning supplies a linear code to the same interface, the same recursive approach can be applied to reduce its proof size and verifier time to $O(\log^2 N)$ while retaining Lightning's faster encoder. Developing and optimizing this combined PCS is left to future work.

Reachable relative distance. The absence of a Lightning configuration at target relative distance 0.040 is not an intrinsic limitation of the Lightning construc-

tion. In this paper, we instantiate Lightning only with Brakedown base codes of relative distance at most 0.070. The distance guarantee for Lightning depends on the parameters of its base code, so a higher-distance base code can support substantially larger Lightning relative distances. For example, using a Reed–Solomon base code would allow Lightning to reach much larger relative distances, such as 0.06. We use Brakedown as the base code here because it has linear-time encoding: combined with Lightning's linear-time sketch, this preserves an $O(n)$ encoder. A Reed–Solomon-based instantiation would instead inherit its $O(n \log n)$ encoding cost.

Acknowledgements

GPT-5 was used to assist with rephrasing and improving the clarity of the manuscript.

References

1. Albrecht, M.R., Fenzi, G., Lapiha, O., Nguyen, N.K.: Slap: Succinct lattice-based polynomial commitments from standard assumptions. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 90–119. Springer Nature Switzerland, Cham (2024)
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 2017 acm sigsac conference on computer and communications security. pp. 2087–2104 (2017)
3. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: Whir: Reed–solomon proximity testing with super-fast verification. In: Fehr, S., Fouque, P.A. (eds.) Advances in Cryptology – EUROCRYPT 2025. pp. 214–243. Springer Nature Switzerland, Cham (2025)
4. Arun, A., Setty, S., Thaler, J.: Jolt: Snarks for virtual machines via lookups. In: Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part VI (2024)
5. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for r1cs. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38. pp. 103–128. Springer (2019)
6. Block, A.R., Fang, Z., Katz, J., Thaler, J., Waldner, H., Zhang, Y.: Field-agnostic snarks from expand-accumulate codes. In: Annual International Cryptology Conference. pp. 276–307. Springer (2024)
7. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Advances in Cryptology – ASIACRYPT 2017 (2017)
8. Bootle, J., Chiesa, A., Groth, J.: Linear-time arguments with sublinear verification from tensor codes. In: Theory of Cryptography Conference. pp. 19–46. Springer (2020)
9. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated pseudorandomness from expand-accumulate codes. In: Annual International Cryptology Conference. pp. 603–633. Springer (2022)
10. Brehm, M., Chen, B., Fisch, B., Resch, N., Rothblum, R.D., Zeilberger, H.: Blaze: Fast snarks from interleaved raa codes. In: Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part IV. p. 123–152. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-91134-7_5, https://doi.org/10.1007/978-3-031-91134-7_5
11. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from dark compilers. In: Advances in Cryptology – EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I. p. 677–706. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-45721-1_24, https://doi.org/10.1007/978-3-030-45721-1_24
12. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 499–530. Springer (2023)

13. Christ, M., Gunn, S.: Pseudorandom error-correcting codes. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024. pp. 325–347. Springer Nature Switzerland, Cham (2024)
14. Comtet, L.: Advanced Combinatorics: The Art of Finite and Infinite Expansions. D. Reidel Publishing Company, Dordrecht (1974)
15. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 90–112 (2012)
16. Couteau, G., Rindal, P., Raghuraman, S.: Silver: silent vole and oblivious transfer from hardness of decoding structured ldpc codes. In: Annual International Cryptology Conference. pp. 502–534. Springer (2021)
17. Divsalar, D., Jin, H., McEliece, R.J.: Coding theorems for" turbo-like" codes. In: Proceedings of the annual Allerton Conference on Communication control and Computing. vol. 36, pp. 201–210. University Of Illinois (1998)
18. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive (2020)
19. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive (2019)
20. Gallager, R.: Low-density parity-check codes. IRE Transactions on Information Theory **8**(1), 21–28 (1962). https://doi.org/10.1109/TIT.1962.1057683
21. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for r1cs. In: Annual International Cryptology Conference. pp. 193–226. Springer (2023)
22. Gurkan, K., Novakovic, A., Rothblum, R.D.: Bolt: Faster SNARKs from sketched codes. Cryptology ePrint Archive, Report 2026/310 (2026), https://eprint.iacr.org/2026/310
23. Haböck, U.: Brakedown's expander code. Cryptology ePrint Archive (2023)
24. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 177–194. Springer (2010)
25. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography. pp. 1–34. Springer International Publishing, Cham (2021)
26. Liu, T., Xie, X., Zhang, Y.: Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2968–2985 (2021)
27. Liu, T., Zhang, Z., Zhang, Y., Hu, W., Zhang, Y.: Ceno: Non-uniform, segment and parallel zero-knowledge virtual machine. J. Cryptol. **38**(2) (Jan 2025). https://doi.org/10.1007/s00145-024-09533-2, https://doi.org/10.1007/s00145-024-09533-2
28. Maram, D., Malvai, H., Zhang, F., Jean-Louis, N., Frolov, A., Kell, T., Lobban, T., Moy, C., Juels, A., Miller, A.: Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1348–1366 (2021). https://doi.org/10.1109/SP40001.2021.00038
29. Nair, V., Sharma, A., Thankey, B.: Brakingbase - a linear prover, poly-logarithmic verifier, field agnostic polynomial commitment scheme. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 427–455. Springer Nature Singapore, Singapore (2026)

30. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Theory of Cryptography Conference. pp. 222–242. Springer (2013)
31. Qu, W., Guo, Y., Xuan, Z., Liu, X., Zhang, J.: Celer: A lookup argument for large-scale queries. In: Annual International Cryptology Conference. pp. 243–266. Springer (2026)
32. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. pp. 459–474. IEEE (2014)
33. Setty, S.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Annual International Cryptology Conference. pp. 704–737. Springer (2020)
34. Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. In: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. pp. 388–397 (1995)
35. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zksnarks without trusted setup. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 926–943. IEEE (2018)
36. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Annual International Cryptology Conference. pp. 299–328. Springer (2022)
37. Zeilberger, H., Chen, B., Fisch, B.: Basefold: efficient field-agnostic polynomial commitment schemes from foldable codes. In: Annual International Cryptology Conference. pp. 138–169. Springer (2024)
38. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 859–876. IEEE (2020)

A Proof of Theorem 2

Proof. We upper bound the probability by a union bound over the set size s and then apply Lemma 3. Let $s_{\max} := \lfloor 0.98n/C \rfloor$. Then

$$\begin{aligned}\Pr\left[\exists\, S \subseteq [n] : |S| \le s_{\max},\, U_A(S) = \emptyset\right] &\le \sum_{s=2}^{s_{\max}} \Pr\left[\exists\, S \subseteq [n] : |S| = s,\, U_A(S) = \emptyset\right] \\ &\le \sum_{s=2}^{s_{\max}} \left(\frac{Cs}{n}\right)^{s(D/2-1)}. \end{aligned} \tag{11}$$

Define $F(s) := (Cs/n)^{s(D/2-1)}$. Treating s as real,

$$\frac{\mathrm{d}}{\mathrm{d}s} \ln F(s) = \left(\frac{D}{2} - 1\right)\left(\ln\left(\frac{Cs}{n}\right) + 1\right).$$

Thus F decreases on $[2, n/(Ce)]$ and increases on $[n/(Ce), 0.98n/C]$, so its maximum on the latter full interval is at an endpoint. The endpoint values are

$$F(2) = \left(\frac{2C}{n}\right)^{D-2}, \qquad F\left(\frac{0.98n}{C}\right) = 0.98^{(D/2-1)0.98n/C}.$$

Using $s_{\max} \le 0.98n/C$ in (11) therefore gives

$$\begin{aligned}&\Pr\left[\exists\, S \subseteq [n] : |S| \le s_{\max},\, U_A(S) = \emptyset\right] \\ &\quad \le \max\left\{ \frac{0.98n}{C}\left(\frac{2C}{n}\right)^{D-2}, \frac{0.98n}{C}\, 0.98^{(D/2-1)0.98n/C} \right\}.\end{aligned}$$

For fixed α, D, the second term is $o(1/\mathsf{poly}(n))$, while the first is $O(n^{3-D})$. This proves the theorem.