Interactive Proofs for Batch Polynomial Evaluation

Gal Arnon
galarnon42@gmail.com
Bocconi University

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Eylon Yogev
eylon.yogev@biu.ac.il
Bar-Ilan University

March 5, 2026

Abstract

Polynomials are a fundamental mathematical object underlying virtually all of theoretical computer science. In proof systems, a common task for the verifier is to evaluate a polynomial of degree d at m distinct points. The best known algorithm for this problem performs $O((m + d) \cdot \log^2(m + d))$ field operations.

We present a concretely efficient MA protocol for this problem in which the verifier runs in *linear time*: the prover sends a single message consisting of $d - 1$ field elements, and the verifier performs only $O(m + d)$ field operations. We further extend our protocol to handle the more general setting of evaluating multiple polynomials at multiple points, and for this problem, we construct an AMA protocol.

Our protocols improve the verifier time in several interactive proofs. Most notably are the sumcheck protocol over a large summation domain and protocols that rely on polynomial quotienting. In particular, by a straightforward application of our results, we reduce the verifier's runtime in the STIR protocol (CRYPTO 2024) to match that of WHIR (EUROCRYPT 2025), despite WHIR being highly optimized for verification time.

As an additional application, we show that any univariate polynomial commitment schemes (PCS) can be transformed, in a black-box manner, into a new scheme that efficiently supports batch openings at multiple points. In particular, opening m points incurs only a constant overhead compared to opening a single point.

Keywords: interactive proofs; polynomial evaluation

Contents

1 Introduction **1**
1.1 Our results . . . 2
1.2 Application: batching polynomial commitment schemes . . . 3
1.3 Comparison with prior work . . . 4

2 Preliminaries **5**
2.1 Zero multiplicities of polynomials . . . 5
2.2 Interactive proofs and PIOPs . . . 5

3 Zero multiplicity distance lemma **7**

4 Protocols for batch polynomial evaluation **10**
4.1 Proof of Theorem 4.2 . . . 11
4.2 Proof of Theorem 4.3 . . . 12

5 Query reduction for PIOPs **16**

6 Improving the verifier time of STIR **20**

Acknowledgments **22**

References **22**

1 Introduction

Polynomials play a central role in the design of cryptographic protocols, where they are used as the algebraic backbone of numerous primitives and constructions. These include important notions such as error-correcting codes, secret sharing schemes, commitment schemes, probabilistically checkable proofs, interactive proofs, and succinct non-interactive arguments. In particular, the algebraic structure of polynomials is at the heart of the efficiency and soundness of many modern proof systems where polynomials are used to encode computation, and their algebraic structure enables the construction of crucial tools such as consistency checks via low-degree testing, interpolation, and evaluation arguments.

As a result, the task of verifying polynomial evaluations—i.e., checking that one or more claimed values correspond to the evaluation of a specific low-degree polynomial at a given set of points—is ubiquitous across interactive and non-interactive proof systems. This task is especially prominent in polynomial commitment schemes and related primitives, where a prover commits to a polynomial and later proves evaluations at points of the verifier's choosing.

Batch polynomial evaluation. In this work, we study the problem $\mathsf{PolyEval}_{\mathbb{F}}$ of batch verification of polynomial evaluation claims over a field $\mathbb{F}$. Given a list of m pairs (x_i, y_i), the verifier must efficiently check that $p(x_i) = y_i$ for all i, where p is a degree d polynomial provided to the verifier in some form (we later give a generalization that considers also the evaluation of multiple polynomials). We usually think of m as being much larger than d. The $\mathsf{PolyEval}_{\mathbb{F}}$ problem arises in various cryptographic protocols, particularly where an untrusted prover is involved. Notable applications include SNARK proof aggregation, the construction of lookup arguments, and protocols leveraging the sumcheck or multilinear extension paradigms (e.g., [LFKN92; Sha92; GKR15; GR17; GR18; RR20; ACFY24]).

A straightforward approach uses Horner's method [BFST03], which evaluates a degree d polynomial at a single point using d field multiplications and d additions. Applying Horner's method independently to each of the m evaluation points results in a total cost of $O(m \cdot d)$ field operations. While Horner's method is optimal for evaluating a single point, more efficient algorithms exist for the multipoint setting.

A classical result in algebraic complexity (see [GG13, Chapter 10] and [Fid72]) is an algorithm that performs $O\left((d+m) \cdot \log^2(d+m)\right)$ field operations. However, upon investigation, the hidden constant in the big-O notation is large in practice, limiting the practicality of this approach for moderate input sizes. Alternatively, the Barycentric interpolation method gives a linear time algorithm of $O(d+m)$, but with a quadratic $O(m^2)$ preprocessing phase (that depends on the points and not the polynomials) [Tre04]. This method is employed across various components of the Ethereum network [But23].

When the evaluation points lie in a structured domain (e.g., FFT-friendly sets), even faster algorithms are possible. However, this paper focuses on the general case of arbitrary evaluation points—a necessity in many applications, for example, when the evaluation points are chosen uniformly at random. As the $\mathsf{PolyEval}_{\mathbb{F}}$ verification problem frequently arises in the context of interactive proofs, we ask:

Can $\mathsf{PolyEval}_{\mathbb{F}}$ *be decided more efficiently with the help of an untrusted prover?*

In this work, we answer this question in the affirmative.

1.1 Our results

We give an efficient single-message protocol for $\mathsf{PolyEval}_{\mathbb{F}}$ where the verifier runs in strictly linear time: it evaluates the polynomial p at a *single* location and, additionally, makes a small number of field operations.

Theorem 1 (informal, Theorem 4.2)**.** *There is an MA proof for* $\mathsf{PolyEval}_{\mathbb{F}}$ *where:*

- *the soundness error is* $\frac{m+d}{|\mathbb{F}|-m}$,
- *the prover sends a single degree* $d-1$ *polynomial and makes* $O(d \cdot m)$ *field operations; and*
- *the verifier makes* $O(d+m)$ *field operations. More precisely, the verifier makes* $2d+3m$ *multiplications and additions and a single inversion.*

The above theorem immediately extends to speeding up polynomial *interpolation*, where the goal is to compute the coefficients of the unique low-degree univariate polynomial that passes through a given set of points. In this setting, the prover can send the interpolated polynomial in coefficient form and then prove to the verifier that it evaluates correctly at all the specified points.

Multiple polynomials and multiple points. We also consider the problem of batch polynomial evaluation with multiple polynomials as well as multiple evaluation points.

In the problem $\mathsf{MultPolyEval}_{\mathbb{F}}$, given degree d polynomials $p_1, \ldots, p_\ell$ and a list of m tuples of the form $(x_i, y_{i,1}, \ldots, y_{i,\ell})$,[1] we check that for all i and j it holds that $p_i(x_j) = y_{i,j}$. Applying Theorem 1 for each of the ℓ polynomials would incur an ℓ multiplicative overhead to the communication. Our next theorem shows that these proofs can be batched together at the cost of a single verifier message:

Theorem 2 (informal, Theorem 4.3)**.** *There is an AMA protocol for* $\mathsf{MultPolyEval}_{\mathbb{F}}$ *where:*

- *the round-by-round soundness*[2] *error is* $\frac{\ell+m+d}{|\mathbb{F}|-m}$,
- *the prover sends a single degree* $d-1$ *polynomial and makes* $O(d \cdot m + d \cdot \ell)$ *field operations; and*
- *the verifier makes* $O(m \cdot \ell + d \cdot \ell)$ *field operations. More precisely, the verifier makes* $d\ell + 5m\ell$ *multiplications,* $d\ell + 4m\ell$ *additions, and a single inversion.*

Large fields. The soundness errors of Theorem 1 and Theorem 2 require the size of the field $|\mathbb{F}|$ to be large relative to the size of the input polynomials. Similarly to prior work, the requirement can be relaxed by sampling challenges in the protocols from a sufficiently large extension field of $\mathbb{F}$.

Immediate applications. Our protocols allow for optimizing the verifier in many protocols where the verifier needs to evaluate polynomials. We describe two main such applications.

- *The sumcheck protocol.* The sumcheck protocol [LFKN92] is an influential protocol used both in theory and practice for proving claims of the form

$$\sum_{x_1,\ldots,x_v \in H} p(x_1, \ldots, x_v) = \gamma \ ,$$

  for a v-variate polynomial p of individual degree d. During the protocol, the verifier must check v claims of the form $\sum_{x \in H} q_i(x) = \gamma_i$ where q_i is a degree d univariate polynomial. This step is typically implemented naively in time $\Omega(v \cdot d \cdot |H|)$.

[1] Actually, we can generalize this further so that not all polynomials must take part in each evaluation point and the polynomials are not required to have the same degree.

[2] Round-by-round soundness is a stronger variant of soundness, which is used when making protocols non-interactive using the Fiat–Shamir heuristic.

Using Theorem 2, the running time is reduced to $O(v \cdot d + v \cdot |H|)$ at the cost of an additional round of communication with $v \cdot |H| + d - 1$ field elements.

- *Quotienting.* Polynomial quotienting is a technique used in practice by many protocols, such as STIR [ACFY24] and ARC [BMNW24]. At a high level, in these protocols, the verifier must evaluate at many points a function

$$f(X) = \frac{g(X) - \mathsf{Ans}(X)}{V(X)} \ ,$$

where g is an oracle function, and Ans and V are degree d polynomials. This computation is often an efficiency bottleneck in the running time of the verifier. Indeed, as a result, the recent WHIR protocol [ACFY25] was developed, in part, to improve the verification time of STIR by avoiding quotient computations on the verifier's side.

Our protocol can be used to directly improve the computation of the quotient, and thus the verifier complexity. We adapt STIR using this technique, resulting in the STIR-SHAKE protocol, which reduces the verifier's running time to asymptotically match that of WHIR. See Section 6 for a more in-depth discussion and analysis.

Immediate generalizations. The techniques in our protocol directly yield a MA protocol to check whether, for a given list of polynomials $u_1, \ldots, u_m$ and $w_1, \ldots, w_m$ (such that the zeros of w_i are pairwise disjoint) the zeros of w_i are contained within the zeros of u_i *up to multiplicities*. The verifier in the generalized protocol performs a single evaluation of each of the u_i, w_i polynomials, an evaluation of a single additional polynomial and $O(m)$ additional field operations. We refer the reader to Section 3 for details.

1.2 Application: batching polynomial commitment schemes

The protocols underlying Theorem 1 and Theorem 2 have a useful structure: they can be interpreted as *polynomial IOPs* (PIOP). In a PIOP, the prover's messages are bounded-degree polynomials, and the verifier queries these polynomials via evaluation queries. Polynomial IOPs [RRR16; CHMMVW20; BFS20] are a fundamental building block of most modern SNARGs. By utilizing this fact about our IPs, we show a generic compiler for reducing the query complexity of any PIOP:

Theorem 3 (informal, Theorem 5.1)**.** *Let Π be a k-round PIOP for a relation $\mathcal{R}$. Then there is a $(k+1)$-round PIOP Π' where the prover sends a single polynomial more than Π, and the verifier makes exactly one query to every polynomial sent by the prover in Π'.*

Theorem 3 has a surprising consequence for (univariate) polynomial commitment schemes. A polynomial commitment scheme (PCS) [KZG10; PST13] is a cryptographic primitive that enables a sender to succinctly commit to a polynomial and then subsequently succinctly open desired evaluations of the committed polynomial. PCS are commonly used to compile PIOPs into arguments. In this context, a highly desirable property of a PCS is "batch opening", where the sender succinctly opens a large set of points (with an opening size that is independent of the number of points). However, not all PCS schemes support batch opening, and often designing a PCS with such property is significantly harder than designing a non-batchable PCS (this was raised in [AFLN24] in the context of batching openings of lattice-based PCSs [FMN24; AFLN24; NS24; CMNW24]).

When compiling PIOPs into arguments, *designing a PCS with batch openings is not necessary*, as the underlying PIOP can be first transformed into a PIOP where each polynomial sent by the prover is queried exactly once by Theorem 3, and then compiled with a PCS. Since the PIOP queries each polynomial exactly once, no batching property of the PCS is required.

1.3 Comparison with prior work

The efficient batching of polynomial commitments and their openings is a widely studied notion in constructing succinct arguments from PIOPs. We present and compare our results to two main examples: Halo Infinite [BDFG21] in the univariate setting, and Hyperplonk [CBBZ23] in the multivariate setting.

- *Univariate.* [BDFG21] considers batching univariate polynomial commitments in the case where the commitments are additively homomorphic (for example, if the commitments are elements of an abelian group) or possesses a linear combination scheme (i.e., it is possible from ℓ commitments and ℓ coefficients to compute a commitment to the linear combination of the committed values with the specified coefficients). Using the language introduced in the recent work [BDRNT25], it yields a version of Theorem 3 for *homomorphic PIOPs*.

  In contrast, our protocols require *no assumption* on the underlying PIOP (and, consequently, on the underlying polynomial commitment scheme). Further, our protocol requires fewer rounds of interaction.

  Taking as an example batching the evaluation of a single polynomial at many points (the setting of Theorem 4.2), our protocols yield an MA proof for batch evaluation of a single polynomial at multiple points. In contrast, the protocol of [BDFG21] requires an additional round of interaction.

- *Multivariate.* [CBBZ23; RR22] deal with reducing query complexity of *multivariate/multilinear PIOPs*. While the problem statement is similar to the univariate case, batching multilinear polynomial evaluations uses significantly different techniques. In particular, all known techniques to solve this problem utilize the multivariate sumcheck protocol [LFKN92], which significantly increases round complexity (typically increasing round complexity to at least $O(m)$ where m is the number of variables) compared to solutions for the univariate case. Our protocol (for univariate polynomials) does not suffer from such an overhead. On the other hand, multilinear batching technique are typically more efficient for the prover (as it needs to perform only a linear amount of field operations). We do not know how to adapt our techniques to achieve linear time complexity, nor how to use them to reduce the round complexity of multilinear polynomial batching, and consider them interesting open problems.

2 Preliminaries

We define objects and notation that we use in this paper.

- For $n \in \mathbb{N}$ we let $[n] \coloneqq \{1, \ldots, n\}$.
- For two functions $f\colon \mathcal{L} \to \mathbb{F}$ and $g\colon \mathcal{D} \to \mathbb{F}$, $\Delta(f, g)$ denotes the fraction of points in $\mathcal{L} \cap \mathcal{D}$ in which they disagree. For a set $\mathcal{S} \subseteq \mathbb{F}^{\mathcal{D}}$, $\Delta(f, \mathcal{S}) \coloneqq \min_{h \in \mathcal{S}} \Delta(f, h)$.

2.1 Zero multiplicities of polynomials

We give definitions regarding the zeroes of polynomials and their multiplicities.

Definition 2.1. *Let $p \in \mathbb{F}[X]$ and $z \in \mathbb{F}$. We define the following:*
- $\mathsf{zeros}(p) \coloneqq \{x \in \mathbb{F} \ : \ p(x) = 0\}$,
- $\mathsf{mult}(p, z) \coloneqq \max\{t \in \mathbb{N} \ : \ (X - z)^t \mid p(X)\}$,
- $\mathsf{zmult}(p) \coloneqq \{(z, t) \ : \ z \in \mathsf{zeros}(p) \wedge t = \mathsf{mult}(p, z)\}$.

Additionally, we write

$$\mathsf{zmult}(p) \sqsubseteq \mathsf{zmult}(q)$$

if for every $(z, t) \in \mathsf{zmult}(p)$ there exists t' such that $(z, t') \in \mathsf{zmult}(q)$ and $t' \geq t$.

We recall the following fact on the multiplicities of the product of polynomials.

Fact 2.2. *For every $z \in \mathbb{F}$ and $p, q \in \mathbb{F}[X]$, $\mathsf{mult}(p \cdot q, z) = \mathsf{mult}(p, z) + \mathsf{mult}(q, z)$.*

2.2 Interactive proofs and PIOPs

A (public-coin) *interactive proof (IP)* for a language L is a tuple of interactive algorithms $(\mathbf{P}, \mathbf{V})$ that works as follows: $\mathbf{P}$ and $\mathbf{V}$ receive as input an instance $\mathbb{x}$ and then interact over k rounds, where in each round $i \in [\mathsf{k}]$ the prover sends a message m_i and the verifier sends a random message α_i. We require the following properties.

- **Completeness.** For every $\mathbb{x} \in L$,

$$\Pr_{\alpha_1, \ldots, \alpha_{\mathsf{k}}} \left[\mathbf{V}(\mathbb{x}, \alpha_1, m_1, \ldots, \alpha_{\mathsf{k}}, m_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{l} m_1 \leftarrow \mathbf{P}(\mathbb{x}) \\ \vdots \\ m_{\mathsf{k}} \leftarrow \mathbf{P}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}) \end{array} \right] = 1 \ .$$

- **Soundness with error β.** For every $\mathbb{x} \notin L$ and unbounded malicious prover $\tilde{\mathbf{P}}$,

$$\Pr_{\alpha_1, \ldots, \alpha_{\mathsf{k}}} \left[\mathbf{V}(\mathbb{x}, \alpha_1, m_1, \ldots, \alpha_{\mathsf{k}}, m_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{l} m_1 \leftarrow \tilde{\mathbf{P}}(\alpha_1) \\ \vdots \\ m_{\mathsf{k}} \leftarrow \tilde{\mathbf{P}}(\alpha_1, \ldots, \alpha_{\mathsf{k}}) \end{array} \right] \leq \beta(\mathbb{x}) \ .$$

MA and AMA. An IP is an *MA proof* if $\mathsf{k} = 1$ in the above definition (the interaction consists of a single prover message followed by a single verifier message). An IP is an *AMA proof* if $\mathsf{k} = 2$ and m_1 is empty in the above definition (the interaction consists of a verifier message, a prover message, and a verifier message).

PIOP. A *polynomial IOP* is an IP where the (honest and malicious) prover is restricted to sending only polynomials (over field and of degree determined as part of the definition of the IP), and the verifier is restricted to reading the prover's messages only by evaluating these polynomials on points in the field. In a PIOP, in addition to running times of the parties, we keep track of the number of polynomials sent, their degrees, and the number of evaluations (queries) made by the verifier.

Round-by-round soundness. A *state function* State for an IP $(\mathbf{P}, \mathbf{V})$ has the following structure for any instance $\mathbb{x}$ and partial transcript tr:

- If $\mathbb{x} \notin L$ and $\mathsf{tr} = \emptyset$ then $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$.
- If tr ends in a verifier message and $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$ then $\mathsf{State}(\mathbb{x}, \mathsf{tr} \| m) = 0$ for every prover message m.
- If tr is a full transcript of $(\mathbf{P}, \mathbf{V})$ then $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = \mathbf{V}(\mathbb{x}, \mathsf{tr})$.

A k-round IP has round-by-round soundness error $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ if there exists a state function State such that for every $\mathbb{x}$ and transcript tr ending in the prover's i-th message where $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$,

$$\Pr_{\alpha_{i+1}} [\mathsf{State}(\mathbb{x}, \mathsf{tr} \| \alpha_{i+1}) = 1] \leq \varepsilon_i \ .$$

We sometimes overload the term, and say that the IP has round-by-round soundness error ε if it has round-by-round soundness error $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ and $\varepsilon \leq \max\{\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}}\}$.

3 Zero multiplicity distance lemma

We state and prove a technical lemma stating that the sum of quotients of polynomials is close to a low-degree polynomial if and only if each numerator vanishes exactly where the denominator does. Consider polynomials $u_1, \ldots, u_m$ and $w_1, \ldots, w_m$ and the quotient function $f \coloneqq \sum_{i \in [m]} u_i / w_i$. The lemma says that if f is sufficiently close to a low-degree polynomial then, for every $i \in [m]$, the zeroes of w_i are zeros of u_i and this holds while respecting multiplicities (if z is a zero of w_i with multiplicity t then z is a zero of u_i with multiplicity at least t).

We use this lemma multiple times by setting u_i to be $p(X) - b_i$ and w_i to be $X - a_i$; in this case, by the lemma, if $p(a_i) \neq b_i$ (in which case $X - a_i$ does not divide $p(X) - b_i$) then the function $\sum_i (p(X) - b_i)/(X - a_i)$ is far from *any* low-degree polynomial.

The lemma is formally stated below using the notation of Definition 2.1. We deem the lemma of independent interest, especially in the context of prover-aided computation (e.g., one can extend our IPs for batch polynomial evaluation to cover any polynomials for which Lemma 3.1 holds).

Lemma 3.1. *Let $(u_1, w_1) \ldots, (u_m, w_m)$ be such that $u_i \in \mathbb{F}^{\leq d_i}[X]$ and $w_i \in \mathbb{F}^{\leq e_i}[X]$, and assume that the sets $(\mathsf{zeros}(w_i))_{i \in [m]}$ are pairwise disjoint. Define $\mathcal{Z} \coloneqq \bigcup_i \mathsf{zeros}(w_i)$, $k \coloneqq \max_i \{d_i - e_i\}$, and $s \coloneqq \sum_i e_i$. Let $f \colon \mathbb{F} \setminus \mathcal{Z} \to \mathbb{F}$ be*

$$f(X) \coloneqq \sum_{i \in [m]} \frac{u_i(X)}{w_i(X)} \ .$$

If $\Delta(f, \mathbb{F}^{\leq k}[X]) \leq 1 - \frac{k+s+1}{|\mathbb{F}| - |\mathcal{Z}|}$, then $\mathsf{zmult}(w_i) \sqsubseteq \mathsf{zmult}(u_i)$ for every $i \in [m]$.

Proof. Suppose $\Delta(f, \mathbb{F}^{\leq k}[X]) \leq 1 - \frac{k+s+1}{|\mathbb{F}| - |\mathcal{Z}|}$ and let $v \in \mathbb{F}^{\leq k}[X]$ be the closest polynomial to f (in terms of Hamming distance over $\mathbb{F} \setminus \mathcal{Z}$ and breaking ties arbitrarily). Letting $S \coloneqq \{z \in \mathbb{F} \setminus \mathcal{Z} : f(z) = v(z)\}$ be the agreement domain between f and v, observe that $|S| > k + s$. Define $A(X) \coloneqq \prod_{i \in [m]} w_i(X)$, and note that $A(X) \in \mathbb{F}^{\leq s}[X]$, and that $A(z) = 0$ for every $z \in \mathcal{Z}$. By definition of f, for every $z \in \mathbb{F} \setminus \mathcal{Z}$,

$$f(z) \cdot A(z) = \sum_{i \in [m]} \frac{u_i(z)}{w_i(z)} \cdot \prod_{j \in [m]} w_j(z) = \sum_{i \in [m]} u_i(z) \cdot \prod_{j \neq i} w_j(z) \ .$$

Define $g_1, g_2 \in \mathbb{F}^{\leq k+s}[X]$ as follows:

$$\begin{aligned} g_1(X) &\coloneqq v(X) \cdot A(X) \ , \\ g_2(X) &\coloneqq \sum_{i \in [m]} u_i(X) \cdot \prod_{j \neq i} w_j(X) \ . \end{aligned}$$

For every $z \in S$, $g_1(z) = v(z) \cdot A(z) = f(z) \cdot A(z) = g_2(z)$. That is, g_1 and g_2 agree on all points in S, and have degree $\leq k + s$. Since $|S| > k + s$, g_1 and g_2 are the same polynomial. Consider $(z_\ell, t_\ell) \in \mathsf{zmult}(w_\ell)$. Then, $w_\ell(X) = (X - z_\ell)^{t_\ell} \cdot z(X)$, and we can write $g_2(X) = (X - z_\ell)^{t_\ell} \cdot s(X) + u_\ell \cdot \prod_{j \neq \ell} w_j(X)$. Further, by Fact 2.2,

$$\mathsf{mult}(g_2, z_\ell) = \mathsf{mult}(g_1, z_\ell) = \mathsf{mult}(v, z_\ell) + \mathsf{mult}(A, z_\ell) \geq t_\ell \ ,$$

which implies that $(X - z_\ell)^{t_\ell}$ divides $u_\ell(X) \cdot \prod_{j \neq \ell} w_j(X)$, and since the sets $\mathsf{zeros}(w_j)$ are pairwise disjoint and thus $z_\ell \notin \mathsf{zeros}(w_j)$ for every $j \neq \ell$, we must have that $(X - z_\ell)^{t_\ell}$ divides u_ℓ and thus $\mathsf{mult}(u_\ell, z_\ell) \geq t_\ell$. □

We restate Lemma 3.1 as a protocol for the following language.

Definition 3.2. *Let $\mathbb{F}$ be a field and let $(w_i)_{i\in[m]}$ be a list of polynomials such that the sets $(\mathsf{zeros}(w_i))_{i\in[m]}$ are pairwise disjoint. We define the following language:*

$$\mathsf{Mult}_{\mathbb{F}}[w_1,\ldots,w_m] \coloneqq \left\{ ((d_i,u_i))_{i\in[m]} \;\middle|\; \begin{array}{l} \forall\, i\in[m]: \\ u_i \in \mathbb{F}^{<d_i}[X] \;\wedge\; \mathsf{zmult}(w_i) \sqsubseteq \mathsf{zmult}(u_i) \end{array} \right\} .$$

The construction follows almost immediately.

Construction 3.3. Fix polynomials $(w_1,\ldots,w_m)$ such that $w_i \in \mathbb{F}^{<e_i}[X]$ and the sets $(\mathsf{zeros}(w_i))_{i\in[m]}$ are pairwise disjoint. Given input $((d_i,u_i))_{i\in[m]}$, the protocol proceeds as follows:

1. Prover sends a polynomial $q \in \mathbb{F}^{\le k}[X]$ where $k \coloneqq \max\{d_i - e_i\}$. In the honest case

$$q(X) := \sum_{i\in[m]} \frac{u_i(X)}{w_i(X)} .$$

2. Verifier samples a random $z \leftarrow \mathbb{F} \setminus \mathcal{Z}$ where $\mathcal{Z} \coloneqq \bigcup_{i\in[m]} \mathsf{zeros}(w_i)$ and accepts if and only if:

$$q(z) = \sum_{i\in[m]} \frac{u_i(z)}{w_i(z)} .$$

Prover and verifier running times. We analyze the running times of the prover and verifier.

- *Prover running time.* For $i \in [m]$, the prover computes the quotients $q_i(X) = u_i(X)/w_i(X)$. This can be done in $O(d_i \cdot e_i)$ field operations via Horner's method for synthetic division [BFST03] or in $O(d_i \cdot \log d_i + e_i \cdot \log e_i)$ field operations via fast Fourier transformations. Obtaining the final quotient can then be done in $O(m \cdot k)$ field operations. The total prover runtime is then

$$O\left(m\cdot k + \sum_{i\in[m]} \min\{d_i\cdot e_i,\, d_i \log d_i + e_i \log e_i\} \right) .$$

- *Verifier running time.* The verifier evaluates $u_1,\ldots,u_m,\ w_1,\ldots,w_m$ and q at a single point and performs m divisions and $3m$ additions. Using the batch inversion method of [Mon87], the divisions can be replaced with $3(m-1)$ multiplications and a single inversion. In total we get:
  - Evaluations: $2m$
  - Additions: $3m$
  - Multiplications: $3m$
  - Inversions: 1

Completeness and soundness. We prove completeness and soundness of Construction 3.3.

Lemma 3.4. *Construction 3.3 has perfect completeness.*

Proof. Fix $((d_i,u_i))_{i\in[m]} \in \mathsf{Mult}_{\mathbb{F}}[w_1,\ldots,w_m]$. Since $\mathsf{zmult}(w_i) \sqsubseteq \mathsf{zmult}(u_i)$, u_i/w_i is a polynomial of degree $d_i - e_i \le k$. Then, $q = \sum_{i\in[m]} u_i/w_i$ is indeed a polynomial of degree $\le k$, and thus the verifier will always accept. □

Lemma 3.5. *Construction 3.3 has soundness error at most*

$$\frac{k+s+1}{|\mathbb{F}|-|\mathcal{Z}|} \ ,$$

where $k, (e_i)_{i\in[m]}, \mathcal{Z}$ *are as is in the construction and* $s \coloneqq \sum_i e_i$.

Proof. Fix $((d_i,u_i))_{i\in[m]} \notin \mathsf{Mult}_{\mathbb{F}}[w_1,\ldots,w_m]$, let q be the prover's message, and define $f\colon \mathbb{F}\setminus\mathcal{Z}\to\mathbb{F}$ as

$$f(z) \coloneqq \sum_{i\in[m]} \frac{u_i(z)}{w_i(z)} \ .$$

Observe that

$$\Pr\left[\text{Verifier accepts}\right] = 1-\Delta\left(f,q\right) \le 1-\Delta\left(f,\mathbb{F}^{\le k}[X]\right) \ .$$

Since (1) $u_i \in \mathbb{F}^{\le d_i}[X]$, (2) $w_i \in \mathbb{F}^{\le e_i}[X]$, (3) $\mathsf{zeros}(w_i)_{i\in[m]}$ are pairwise disjoint, by applying Lemma 3.1 we have $\Delta\left(f,\mathbb{F}^{\le k}[X]\right) \ge 1-\frac{k+s+1}{|\mathbb{F}|-|\mathcal{Z}|}$, and so

$$\Pr\left[\text{Verifier accepts}\right] \le \frac{k+s+1}{|\mathbb{F}|-|\mathcal{Z}|} \ .$$

□

4 Protocols for batch polynomial evaluation

We use Lemma 3.1 to construct interactive proofs for batch-evaluation of univariate polynomials. In fact, in all of our protocols, the prover sends a single message, which represents a polynomial. The verifier only utilizes this polynomial by evaluating it on a point. Thus, we describe them as PIOPs. Moving to the standard interactive proof model can be done trivially by the prover explicitly sending the polynomials as coefficients, and the verifier can now evaluate them directly.

For the reader's convenience, we have added a description of the verifier's running times in the IP model in the efficiency analysis section of each protocol.

We begin by defining the languages for which we construct PIOPs:

Definition 4.1. *Let $\mathbb{F}$ be a field. We define the following languages.*

- ***Single polynomial.***

$$\mathsf{PolyEval}_{\mathbb{F}} := \left\{ (\mathcal{S}, d, p, (b_x)_{x\in\mathcal{S}}) \;\middle|\; \begin{array}{l} \mathcal{S} \subseteq \mathbb{F},\ p \in \mathbb{F}^{\leq d}[X] \\ \forall x \in \mathcal{S}\colon b_x \in \mathbb{F} \ \wedge\ p(x) = b_x \end{array} \right\} .$$

- ***Multiple polynomials.***

$$\mathsf{MultPolyEval}_{\mathbb{F}} := \left\{ \left(\begin{array}{c} \mathcal{S}, \\ \{(d_{x,i}, p_{x,i}, b_{x,i})\}_{x\in\mathcal{S}, i\in[n_x]} \end{array} \right) \;\middle|\; \begin{array}{l} \mathcal{S} \subseteq \mathbb{F} \\ \forall x \in \mathcal{S}, i \in [n_x] : \left(\begin{array}{c} p_{x,i} \in \mathbb{F}^{\leq d_{x,i}}[X],\ b_{x,i} \in \mathbb{F} \\ p_{x,i}(x) = b_{x,i} \end{array} \right) \end{array} \right\} .$$

We can now state our theorems: an MA protocol for $\mathsf{PolyEval}_{\mathbb{F}}$ and AMA protocol for $\mathsf{MultPolyEval}_{\mathbb{F}}$.

Theorem 4.2. *Given any field $\mathbb{F}$, there is an MA PIOP for $\mathsf{PolyEval}_{\mathbb{F}}$ with the following properties.*

- *Soundness error: $\frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}$.*
- *Polynomials sent: one polynomial of degree $d-1$.*
- *Prover efficiency: $O(d \cdot |\mathcal{S}|)$ field operations.*
- *Verifier randomness: $O(\log |\mathbb{F}|)$ bits.*
- *Verifier efficiency: One evaluation of p and of the polynomial sent by the prover, $3|\mathcal{S}|$ field additions and multiplications, and one field inversion.*

Theorem 4.3. *Given any field $\mathbb{F}$, there is an AMA PIOP for $\mathsf{MultPolyEval}_{\mathbb{F}}$ with the following properties.*

- *Round-by-round soundness error: $\left(\frac{\max_{x\in\mathcal{S}} n_x - 1}{|\mathbb{F}|}, \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}\right)$ where $d := \max_{x\in\mathcal{S}, i\in[n_x]} d_{x,i}$.*
- *Prover message length: d field elements.*
- *Prover efficiency: $O(d \cdot |\mathcal{S}| + \sum_{s\in\mathcal{S}, i\in[n_x]} d_{x,i})$ field operations.*
- *Verifier randomness: $O(\log |\mathbb{F}|)$ bits.*
- *Verifier efficiency: One evaluation of each $p_{x,i}$ (all at the same point) and of the polynomial sent by the prover, and, additionally, $4n$ additions, $5n$ multiplications, and a single inversion (where $n := \sum_{x\in\mathcal{S}} n_x$).*

4.1 Proof of Theorem 4.2

Construction 4.4. Given input $(\mathcal{S}, d, p, (b_x)_{x\in\mathcal{S}})$, the protocol proceeds as follows:

1. Prover sends a polynomial $q \in \mathbb{F}^{\leq d-1}[X]$. In the honest case

$$q(X) := \sum_{x\in\mathcal{S}} \frac{p(X) - b_x}{X - x} \ .$$

2. Verifier samples a random $z \leftarrow \mathbb{F} \setminus \mathcal{S}$ and accepts if and only if:

$$q(z) = \sum_{x\in\mathcal{S}} \frac{p(z) - b_x}{z - x} \ .$$

Prover and verifier running times. We analyze the running times of the prover and verifier.

- *Prover running time.* The prover computes the quotients $q_x(X) = (p(X) - b_x)/(X - x)$ for every $x \in \mathcal{S}$, which can be done in $O(d)$ field operations per quotient via Horner's method for synthetic division [BFST03]. Obtaining q then entails summing these $|\mathcal{S}|$ quotients, for a total of $O(d \cdot |\mathcal{S}|)$ field operations.

- *Verifier running time.* The verifier evaluates p and q at a single point and performs $|\mathcal{S}|$ divisions and $3|\mathcal{S}|$ additions. Using Montgomery's batch inversion method [Mon87], the divisions can be replaced with $3(|\mathcal{S}| - 1)$ multiplications and a single inversion. In total we get:
  - Evaluations: 2
  - Additions: $3|\mathcal{S}|$
  - Multiplications: $3|\mathcal{S}|$
  - Inversions: 1

  Alternatively, if the polynomials are given explicitly by their coefficients, then evaluating them can be done using Horner's method (degree many multiplications and additions) [BFST03]. In this case, the total number of operations that the verifier performs is:
  - Additions: $d + d - 1 + 3|\mathcal{S}| \leq 2d + 3|\mathcal{S}|$
  - Multiplications: $d + d - 1 + 3(|\mathcal{S}| - 1) \leq 2d + 3|\mathcal{S}|$
  - Inversions: 1

Completeness. Fix $(\mathcal{S}, d, p, (b_x)_{x\in\mathcal{S}}) \in \mathsf{PolyEval}_{\mathbb{F}}$. We show that the verifier always accepts when interacting with the honest prover. More specifically, we show that the polynomial

$$q(X) := \sum_{x\in\mathcal{S}} \frac{p(X) - b_x}{X - x} \ ,$$

as defined in the protocol has degree smaller than $d - 1$, and so the honest prover can send it to the verifier. Given that the polynomial $\sum_{x\in\mathcal{S}} \frac{p(X)-b_x}{X-x}$ is sent to the verifier, its check will pass with probability 1.

Fix $x \in \mathcal{S}$. Recall that $p(x) = b_x$, and so $p \in \mathbb{F}^{\leq d}[X]$ can be rewritten as

$$p(X) = (X - x) \cdot q_x(X) + b_x \ ,$$

where $q_x \in \mathbb{F}^{\leq d-1}[X]$. Thus,

$$q(X) = \sum_{x \in \mathcal{S}} \frac{p(X) - b_x}{X - x} = \sum_{x \in \mathcal{S}} \frac{((X - x) \cdot q_x(X) + b_x) - b_x}{X - x} = \sum_{x \in \mathcal{S}} q_x(X) \ ,$$

and so $\mathsf{deg}(q) = \max\{\mathsf{deg}(q_x)\} \leq d - 1$.

Soundness. Fix $(\mathcal{S}, d, p, (b_x)_{x \in \mathcal{S}}) \notin \mathsf{PolyEval}_{\mathbb{F}}$, let q be the prover's message, and define $f \colon \mathbb{F} \setminus \mathcal{S} \to \mathbb{F}$ as

$$f(z) \coloneqq \sum_{x \in \mathcal{S}} \frac{p(z) - b_x}{z - x} \ .$$

Observe that

$$\Pr\left[\text{Verifier accepts}\right] = 1 - \Delta\left(f, q\right) \leq 1 - \Delta\left(f, \mathbb{F}^{\leq d-1}[X]\right) \ .$$

For $x \in \mathcal{S}$ let $u_x(X) \coloneqq p(X) - b_x$ and $w_x(X) \coloneqq X - x$. Since $(\mathcal{S}, p, (b_x)_{x \in \mathcal{S}}) \notin \mathsf{PolyEval}_{\mathbb{F}}$, there exists x' so that $p(x') \neq b_{x'}$. Observe that (1) $u_x \in \mathbb{F}^{\leq d}[X]$, (2) $w_x \in \mathbb{F}^{\leq 1}[X]$, (3) $\mathsf{zeros}(w_x) = \{x\}$ are pairwise distinct, and (4) $\mathsf{zeros}(w_{x'}) \not\subseteq \mathsf{zeros}(u_{x'})$.

By applying Lemma 3.1 we have $\Delta\left(f, \mathbb{F}^{\leq d-1}[X]\right) \geq 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}$, and so

$$\Pr\left[\text{Verifier accepts}\right] \leq \frac{|\mathcal{S}| + d}{|\mathbb{F}| - |\mathcal{S}|} \ .$$

4.2 Proof of Theorem 4.3

Construction 4.5. Fix inputs $\mathcal{S}$, and $\{(d_{x,i}, p_{x,i}, b_{x,i})\}_{x \in \mathcal{S}, i \in [n_x]}$ for $\mathsf{MultPolyEval}_{\mathbb{F}}$.

1. Verifier samples $r \leftarrow \mathbb{F}$ and sends it to the prover.
2. Prover sends a polynomial $q \in \mathbb{F}^{\leq d-1}[X]$ where $d \coloneqq \max_{x \in \mathcal{S}, i \in [n_x]} d_{x,i}$. In the honest case, q defined as follows:
$$q(X) \coloneqq \sum_{x \in \mathcal{S}} \frac{\sum_{i \in [n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})}{X - x} \ .$$
3. Verifier samples a random $z \leftarrow \mathbb{F} \setminus \mathcal{S}$ and accepts if and only if:
$$q(z) = \sum_{x \in \mathcal{S}} \frac{\sum_{i \in [n_x]} r^{i-1} \cdot (p_{x,i}(z) - b_{x,i})}{z - x} \ .$$

Prover and verifier running times. We analyze the prover and verifier running times:

- *Prover running time.* For $x \in \mathcal{S}$ the prover computes the polynomial $t_x(X) = \sum_{i \in [n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})$ in $O(\sum_{s \in \mathcal{S}, i \in [n_x]} d_{x,i})$ field operations. For $x \in \mathcal{S}$, the prover then computes $q_x(X) = t_x(X)/(X - x)$ in $O(d)$ operations per quotient again via Horner's rule for division [BFST03]. Finally, computing q requires $O(|\mathcal{S}| \cdot d)$ field operations. In total, this results in $O(|\mathcal{S}| \cdot d + \sum_{s \in \mathcal{S}, i \in [n_x]} d_{x,i})$ field operations.

- *Verifier running time.* The verifier evaluates each polynomial $\{p_{x,i}\}_{x\in\mathcal{S},\, i\in[n_x]}$ once (note that if a polynomial appears twice in the list of polynomials, it is still evaluated only once), and evaluates q once. It then computes the powers r^i using at most n multiplications (recall that $n = \sum_{x\in\mathcal{S}} n_x$). It then performs $|\mathcal{S}|$ divisions, n multiplications, and $2n + 2|\mathcal{S}|$ additions. Using Montgomery's batch inversion method [Mon87], the divisions can be replaced with $3(|\mathcal{S}|-1)$ multiplications and a single inversion. In total we get:
  - Evaluations: $n + 1$
  - Additions: $2n + 2|\mathcal{S}| \leq 4n$
  - Multiplications: $2n + 3|\mathcal{S}| \leq 5n$
  - Inversions: 1

  Alternatively, if the polynomials are given explicitly by their coefficients, then evaluating them can be done using Horner's method (degree many multiplications and additions) [BFST03]. In this case, the total number of operations that the verifier performs is:
  - Additions: $dn + d - 1 + 2n + 2|\mathcal{S}| \leq (n + 1)d + 4n$
  - Multiplications: $dn + d - 1 + 2n + 3(|\mathcal{S}| - 1) \leq (n + 1)d + 5n$
  - Inversions: 1

Completeness. Fix $(\mathcal{S}, (d_{x,i}, p_{x,i}, b_{x,i})_{x\in\mathcal{S},i\in[n_x]}) \in \mathsf{MultPolyEval}_{\mathbb{F}}$. We show that for every r the polynomial

$$q(X) := \sum_{x\in\mathcal{S}} \frac{\sum_{i\in[n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})}{X - x} \ ,$$

as defined in the protocol has degree bounded by $d-1$, and so can be sent by the prover. Once this is established, then by definition the verifier will accept with probability 1.

For every $x \in \mathcal{S}$ and $p_{x,i} \in \mathbb{F}^{\leq d_{x,i}}[X]$, we have that $p_{x,i}(x) = b_{x,i}$. Thus, $p_{x,i}$ can be rewritten as

$$p_{x,i}(X) \coloneqq (X - x) \cdot q_{x,i}(X) + b_{x,i} \ ,$$

where $q_{x,i} \in \mathbb{F}^{\leq d_{x,i}-1}[X]$. Therefore, for every $r \in \mathbb{F}$ we can rewrite

$$\begin{aligned} q(X) &= \sum_{x\in\mathcal{S}} \frac{\sum_{i\in[n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})}{X - x} \\ &= \sum_{x\in\mathcal{S}} \frac{\sum_{i\in[n_x]} r^{i-1} \cdot (((X - x) \cdot q_{x,i}(X) + b_{x,i}) - b_{x,i})}{X - x} \\ &= \sum_{x\in\mathcal{S}} \sum_{i\in[n_x]} r^{i-1} \cdot q_{x,i}(X) \ . \end{aligned}$$

Therefore $\mathsf{deg}(q) = \max\{\mathsf{deg}(q_{x,i})\} \leq d - 1$.

Round-by-round soundness. We analyze the round-by-round soundness of our protocol. Before describing the state function, we prove a useful claim:

Claim 4.6. *Fix $\mathcal{S} \subseteq \mathbb{F}$, $x \in \mathcal{S}$, and for $i \in [n]$, $p_{x,i} \in \mathbb{F}^{\leq d_{x,i}}[X]$ and $b_{x,i} \in \mathbb{F}$. Set $d \coloneqq \max_{x,i} d_{x,i}$ and define the function $f_r \colon \mathbb{F} \setminus \mathcal{S} \to \mathbb{F}$ as follows*

$$f_r(X) = \sum_{x\in\mathcal{S}} \frac{\sum_{i\in[n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})}{X - x} \ .$$

If for some $y \in \mathcal{S}$ and $i \in [n_y]$ we have that $p_{y,i}(y) \neq b_{y,i}$ then

$$\Pr\left[\Delta(f_r, \mathbb{F}^{\leq d-1}[X]) \leq 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}\right] \leq \frac{\max_{x\in\mathcal{S}} n_x - 1}{|\mathbb{F}|} \ .$$

Proof. Let $y \in \mathcal{S}$ and $i \in [n_y]$ be such that $p_{y,i}(y) \neq b_{y,i}$ Then $g_y(X) \coloneqq \sum_{i\in[n_y]} X^{i-1} \cdot (p_{y,i}(y) - b_{y,i})$ is not the zero polynomial. Noting that $\mathsf{deg}(g_y)) \leq n_y - 1$, we have that

$$\Pr_{r\leftarrow\mathbb{F}}[g_y(r) = 0] \leq \frac{n_y - 1}{|\mathbb{F}|} \ .$$

For $x \in \mathcal{S}$ and $r \in \mathbb{F}$ let $u_x^{(r)}(X) := \sum_{i\in[n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})$ and observe that $g_y(r) = u_y^{(r)}(y)$. Letting $w_x(X) := X - x$, notice that $\mathsf{zeros}(w_x) = \{x\}$ and so $\{\mathsf{zeros}(w_x)\}_{x\in\mathcal{S}}$ are pairwise disjoint. Since $f_r(X) = \sum_{x\in\mathcal{S}} u_x^{(r)}(X)/w_x(X)$, by Lemma 3.1 we have that if the conditions of Claim 4.6 hold then it must be the case that $\mathsf{zeros}(w_x) \subseteq \mathsf{zeros}(u_x^{(r)})$ for every $x \in \mathcal{S}$ and thus the condition must in particular hold for $x = y$ i.e. $\{y\} = \mathsf{zeros}(w_y) \subseteq \mathsf{zeros}(u_y^{(r)})$. In other words, this can only happen if $g_y(r) = u_y^{(r)}(y) = 0$. Then,

$$\Pr_{r\leftarrow\mathbb{F}}\left[\Delta(f_r, \mathbb{F}^{\leq d-1}[X]) \leq 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}\right] = \Pr_{r\leftarrow\mathbb{F}}[g_y(r) = 0] \leq \frac{n_y - 1}{|\mathbb{F}|} \leq \max_{x\in\mathcal{S}} \frac{n_x - 1}{|\mathbb{F}|} \ .$$

□

Lemma 4.7. *The protocol has round-by-round soundness error*

$$(\varepsilon^{\mathsf{rand}}, \varepsilon^{\mathsf{prox}}) = \left(\frac{\max_{x\in\mathcal{S}} n_x - 1}{|\mathbb{F}|}, \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}\right) \ .$$

Proof. We define a state function State, and bound each term individually. Below, we denote the input to the protocol by $\mathbb{x} \coloneqq \left(\mathcal{S}, \{(d_{x,i}, p_{x,i}, b_{x,i})\}_{x\in\mathcal{S}, i\in[n_x]}\right)$.

0. **Initial state.** We set $\mathsf{State}(\mathbb{x}, \emptyset) = 1$ if and only if $\mathbb{x} \in \mathsf{MultPolyEval}_{\mathbb{F}}$.

1. **Bounding** $\varepsilon^{\mathsf{rand}}$. At this stage, tr is empty, and the verifier sends $r \leftarrow \mathbb{F}$.

   - *State function.* Define the function $f_r : \mathbb{F} \setminus \mathcal{S} \to \mathbb{F}$ as follows

   $$f_r(X) := \sum_{x\in\mathcal{S}} \frac{\sum_{i\in[n_x]} r^{i-1} \cdot (p_{x,i}(X) - b_{x,i})}{X - x} \ .$$

   We set $\mathsf{State}(\mathbb{x}, r) = 1$ if and only if

   $$\Delta(f_r, \mathbb{F}^{\leq d-1}[X]) \leq 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|} \ .$$

   - *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \emptyset) = 0$. This implies that there exist $x \in \mathcal{S}$ and $i \in [n_x]$ with $p_{x,i}(x) \neq b_{x,i}$. Then, we are exactly in the setting of Claim 4.6 and

   $$\begin{aligned}\Pr\left[\mathsf{State}(\mathbb{x}, r) = 1 | \mathsf{State}(\mathbb{x}, \emptyset) = 0\right] &= \Pr\left[\Delta(f_r, \mathbb{F}^{\leq d-1}[X]) \leq 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|} \;\middle|\; \mathsf{State}(\mathbb{x}, \emptyset) = 0\right] \\ &\leq \max_{x\in\mathcal{S}} \frac{n_x - 1}{|\mathbb{F}|} \ .\end{aligned}$$

2. **Bounding** $\varepsilon^{\mathsf{prox}}$. At this stage $\mathsf{tr} := (r, q)$ and the verifier sends $z \leftarrow \mathbb{F} \setminus \mathcal{S}$.

   - *State function.* We set $\mathsf{State}(\mathbb{x}, \mathsf{tr} \| z) = 1$ if and only if the verifier accepts given $\mathbb{x}$, tr, and z, i.e. if $q(z) = f_r(z)$.
   - *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$. Then $\Delta(f_r, q) \geq \Delta(f_r, \mathbb{F}^{\leq d-1}[X]) > 1 - \frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}$, and thus the probability that $f_r(z) = q(z)$ for $z \leftarrow \mathbb{F} \setminus \mathcal{S}$ is at most $\frac{|\mathcal{S}|+d}{|\mathbb{F}|-|\mathcal{S}|}$.

□

5 Query reduction for PIOPs

We describe a compiler that, using Theorem 4.3, transforms a PIOP into a PIOP where each proof polynomial is queried exactly once.

Theorem 5.1. *Given a* k*-round PIOP* Π *for a relation* $\mathcal{R}$ *with maximal degree* d *and total query complexity* q *we construct a* $\mathsf{k}+1$ *PIOP* Π' *for* $\mathcal{R}$ *with the following properties:*

- *The prover in* Π' *sends exactly the same messages as* Π *with the addition of* q *field elements and a single polynomial of degree* $d-1$*.*
- *Every polynomial sent by the prover in* Π' *is queries exactly once, all on the same field element.*

Remark 5.2. For generality, we write the protocol in a way that enables the following generalizations of the above theorem:

- *Polynomial IOPs of proximity.* The verifier is given oracle access to part of its input $\mathbb{y}$. We further note that if the implicit input contains polynomials with evaluation access, then they can also be batched into the batch evaluation protocol (these can be considered as an initial prover message).
- *Polynomial IOP for Relations.* The IOP is for a relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{w})\}$ and the honest prover is given the witness $\mathbb{w}$ (in a proof of proximity we would consider $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}), \mathbb{w}\}$). In many use-cases this allows the honest prover to be more efficient.
- *Round-by-round knowledge.* We show that if the original PIOP has round-by-round knowledge with an extractor $\mathbf{E}$, then the new IOP also has round-by-round knowledge soundness.

See [CY24] for formal definitions of all of the above generalizations.

Proof. We detail the transformation and then prove its properties. While our transformation uses Theorem 4.3 directly, for convenience and minor optimizations we describe below the transformation with Theorem 4.3 explicitly written out. Let $\Pi = (\mathbf{P}, \mathbf{V})$ we begin by giving notation to the parameters of Π:

- In round $i \in [\mathsf{k}]$, the prover sends n_i polynomials, where for $j \in [n_i]$ the j-th polynomial $f_{i,j}$ has degree $d_{i,j}$. Let $\mathsf{q}_{i,j}$ be the number of queries made by the verifier to $f_{i,j}$.
- In round i, the verifier sends r_i bits of randomness.
- Let t be the maximal number of polynomials that are evaluated on the same point, and m be the maximal number of distinct points evaluated on any of the polynomials sent.
- Π has round-by-round knowledge soundness error $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ with state function $\mathsf{State}_{\mathsf{poly}}$, extractor $\mathbf{E}$, and extraction time et.
- For simplicity, the protocol described below does not explicitly state non-oracle messages sent by the prover, but these are straightforward to add. l denotes the total number of such field elements sent.

Construction 5.3. We construct a PIOP for $\mathcal{R}$ given a PIOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$.

0. **Inputs:** The honest prover receives as input $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$, and the verifier receives $\mathbb{x}$ as an explicit input and $\mathbb{y}$ as an oracle input.

1. **PIOP interaction phase:** For $i = 1, \ldots, \mathsf{k}$:

   (a) **PIOP prover message:** The prover computes and sends $f_{i,1}, \ldots, f_{i,n_i}$ where $f_{i,j} \in \mathbb{F}^{\leq d_{i,j}}[X]$. In the honest case $(f_{i,1}, \ldots, f_{i,n_i}) := \mathbf{P}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i-1})$.

(b) **PIOP verifier message:** The verifier samples and send $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.

2. **Batching interaction phase:**

(a) **Send query results:** The prover sends an array of field elements $(A_{i,j})_{i\in[\mathsf{k}],j\in[n_i]}$ with $|A_{i,j}| = \mathsf{q}_{i,j}$. In the honest case, the prover simulates the execution of

$$\mathbf{V}^{\mathbb{y},(f_{i,j})_{i\in[\mathsf{k}],j\in[n_i]}}(\mathbb{x},\alpha_1,\ldots,\alpha_{\mathsf{k}}) \ .$$

For every $i \in [\mathsf{k}]$ and $j \in [n_i]$ set:

- $\mathcal{Q}_{i,j} \subseteq \mathbb{F}$ be the set of queries made by $\mathbf{V}$ to $f_{i,j}$.
- $\phi_{i,j} \colon [\mathsf{q}_{i,j}] \to \mathcal{Q}_{i,j}$ be the mapping where $\phi_{i,j}(k)$ returns the k-th query made to $f_{i,j}$;

Then, for $k \in [\mathsf{q}_{i,j}]$, the prover sets $A_{i,j}[k] := f_{i,j}(\phi_{i,j}(k))$.

(b) **Sample combination randomness:** the verifier sends randomness $r \leftarrow \mathbb{F}$.

(c) **Send interpolation polynomial:** the prover sends a polynomial $q \in \mathbb{F}^{\leq d-1}[X]$ where $d := \max_{i,j} d_{i,j}$. In the honest case,

$$q(X) := \sum_{q\in\mathcal{Q}} \frac{\sum_{k\in[t_q]} r^{k-1}\cdot\left(f_{\psi_q(k)}(X) - f_{\psi_q(k)}(q)\right)}{X-q} \ . \tag{1}$$

where:

- $\mathcal{Q} = \bigcup_{i,j} \mathcal{Q}_{i,j}$ is the set of all queries made by the verifier, and
- For $q \in \mathcal{Q}$: $S_q := \{(i,j) : q \in \mathcal{Q}_{i,j}\}$, $t_q := |S_q|$, and $\psi_q : [t_q] \to S_q$ is an arbitrary ordering of S_q.

(d) **Choose proximity randomness:** the verifier determines $\mathcal{Q}$ as before, samples a random $z \leftarrow \mathbb{F} \setminus \mathcal{Q}$ and sends it to the prover.

3. **Verifier decision phase:**

(a) **PIOP decision:** Check that $\mathbf{V}^{\mathbb{y},(f_{i,j})_{i\in[\mathsf{k}],j\in[n_i]}}(\mathbb{x},\alpha_1,\ldots,\alpha_{\mathsf{k}}) = 1$, where the k-th query to $f_{i,j}$ is answered by $A_{i,j}[k]$ and queries to $\mathbb{y}$ are answered by querying $\mathbb{y}$ directly (reject if $\mathbf{V}$ rejects).

(b) **Batching decision:** Letting ψ_q be as before, check that

$$q(z) = \sum_{q\in\mathcal{Q}} \frac{\sum_{k\in[t_q]} r^{k-1}\cdot\left(f_{\psi_q(k)}(z) - A_{\psi_q(k)}[\phi^{-1}(q)]\right)}{z-q} \ ,$$

querying q and $(f_{i,j})_{i,j}$ accordingly.

Complexity measures. The new PIOP has the following parameters:

- *Rounds.* $\mathsf{k}+1$.
- *Polynomials sent.* In round $i \in [\mathsf{k}]$ the prover sends n_i polynomials, where, for $j \in [n_i]$, the function has degree $d_{i,j}$. In round $\mathsf{k}+1$, the prover sends a single polynomial of degree $d-1$.

- *Non-oracle messages.* The prover sends $\mathsf{l}_{\mathsf{poly}}$ field elements in the interaction of Π, and q additional field elements.
- *Evaluation queries.* The verifier makes one evaluation query to each of the $1 + \sum_{i\in[\mathsf{k}]} n_i$ polynomials sent in the protocol.

Round-by-round soundness. We prove that Π' has round-by-round soundness error

$$(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}}, \varepsilon_{\mathsf{com}}, \varepsilon_{\mathsf{prx}}) = \left(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}}, \frac{t-1}{|\mathbb{F}|}, \frac{d+m}{|\mathbb{F}|-m}\right) \ ,$$

with extraction time $O(\mathsf{et})$.

We define a state function State, and bound each term individually.

0. **State function for empty transcript** We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \emptyset) = \mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \emptyset)$.
1. **PIOP verifier message:** At this point of the interaction, the transcript has the form

$$\mathsf{tr} = \begin{pmatrix} ((f_{\ell,1}, \ldots, f_{\ell,m_\ell}), \alpha_\ell)_{\ell<i} \\ (f_{i,1}, \ldots, f_{i,m_i}), \end{pmatrix} \ .$$

   The verifier chooses $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.

   - *State function:* We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|\alpha_i) = \mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|\alpha_i)$.
   - *Bounding the error:* Suppose that

$$\Pr[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|\alpha_i) = 1 | \mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0] > \varepsilon_i$$

     Then, $\Pr[\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|\alpha_i) = 1 | \mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0] > \varepsilon_i$ and by round-by-round knowledge soundness of Π we have $(\mathbb{x}, \mathbb{y}, \mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})) \in \mathcal{R}$.
2. **Combination randomness:** At this point of the interaction, the transcript has the form

$$\mathsf{tr} = \begin{pmatrix} ((f_{i,1}, \ldots, f_{i,m_i}), \alpha_i)_{i\leq\mathsf{k}} \\ (A_{i,j})_{i\in[\mathsf{k}], j\in[m_i]} \end{pmatrix} \ .$$

   The verifier chooses $r \leftarrow \mathbb{F}$.

   - *State function:* We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|r) = 1$ if both of the following hold:
     (a) $\mathbf{V}$ accepts when given access to $\mathbb{y}$ and given query answers according to $A_{i,j}$.
     (b) Letting h_r be the right-hand side of Equation 1,

$$\Delta(h_r, \mathbb{F}^{\leq d-1}[X]) \leq 1 - \frac{d+m}{|\mathbb{F}|-m} \ .$$

   - *Bounding the error:* We show that

$$\Pr[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}\|r) = 1 | \mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0] \leq \frac{\max_{q\in\mathcal{Q}} t_q - 1}{|\mathbb{F}|} = \frac{t-1}{|\mathbb{F}|} \ .$$

     Since the probability is always bounded from above, we do not require the extractor to be able to extract. Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = \mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$. Then, if it holds that for every $i \in \mathsf{k}$, $j \in [n_i]$ and $k \in [\mathsf{q}_{i,j}]$ $A_{i,j}[k] = f_{i,j}(\phi_{i,j}(k))$ the $\mathbf{V}$ can never accept by the round-by-round knowledge soundness of Π, and we are done. Thus, there must exists a triple (i,j,k) such that $A_{i,j}[k] \neq f_{i,j}(\phi_{i,j}(k))$, and we are left to bound the second item in this case. The claimed error follows then directly by applying Claim 4.6.

3. **Proximity randomness:** At this point of the interaction, the transcript has the form

$$\mathsf{tr} = \begin{pmatrix} ((f_{i,1}, \ldots, f_{i,m_i}), \alpha_i)_{i \le \mathsf{k}} \\ (A_{i,j})_{i \in [\mathsf{k}], j \in [m_i]}, r, q \end{pmatrix} .$$

The verifier chooses $z \leftarrow \mathbb{F} \setminus \mathcal{Q}$.

- *State function:* We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| r) = 1$ if both of the following hold:
  (a) $\mathbf{V}$ accepts when given access to $\mathbb{y}$ and given query answers according to $A_{i,j}$.
  (b) $q(z) = f_r(z)$ where f_r is the right hand side of Equation 1 with r fixed.
- *Bounding the error:* Since the probability is always bounded from above, we do not require the extractor to be able to extract. Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$. Then, by definition, $\Delta(f_r, q) \geq \Delta(f_r, \mathbb{F}^{\leq d}[X]) > 1 - \frac{d+m}{|\mathbb{F}|-m}$, and thus the probability that $f_r(z) = q(z)$ for $z \leftarrow \mathbb{F} \setminus \mathcal{Q}$ is at most $\frac{d+m}{|\mathbb{F}|-m}$.

□

6 Improving the verifier time of STIR

Our techniques can be used to improve the verifiers that use quotienting of polynomials. We discuss a concrete example: the STIR protocol [ACFY24]. Specifically, we show how to improve the STIR protocol so to that the verifier time matches that of the WHIR protocol [ACFY25] (which was designed specifically with verification time in mind).

Overview of STIR. The STIR protocol is an interactive oracle proof of proximity[3] for proving that a function $f\colon \mathcal{L} \to \mathbb{F}$ is close to the Reed–Solomon code $\mathsf{RS}[d, \mathcal{L}]$ (i.e., it is close in Hamming weight to the evaluation over $\mathcal{L}$ of a degree d polynomial). Given a security parameter λ and a (small constant) "folding parameter" $k > 2$, the protocol consists of $M = O(\log d)$ iterations where, in iteration i, the prover sends a function g_i, and the verifier has two degree t_i polynomials Ans_i and V_i where $t_i \approx \lambda/i$. The verifier then virtually defines the function

$$f_i(X) \coloneqq \frac{g_i(X) - \mathsf{Ans}_i(X)}{V_i(X)} \ .$$

The verifier then evaluates f_i on $k \cdot t_{i+1}$ points (where $t_{i+1} \approx \lambda/(i+1)$), which reduces to making $k \cdot t_{i+1}$ queries to g_i, and evaluating Ans_i and V_i on t_{i+1} points. It then uses the answers to generate Ans_{i+1}, V_{i+1}, and (in the final round) do a small number of efficient equality tests for final verification.

In [ACFY24], in order to evaluate Ans_i and V_i on all points, the verifier performs roughly $\Omega((t_i + kt_{i-1}) \cdot \log^2(t_i + kt_{i-1}) = \Omega(kt_i \cdot \log^2(kt_i))$ field operations using the classical batch polynomial evaluation algorithm. In fact, in practice the hidden constants in the classical batch polynomial evaluation algorithm are large, and so it is more common to use Horner's method, which yields asymptotic running time $\Omega(kt_i^2)$. Thus, overall, the verifier complexity is $\Omega\left(\sum_{i=1}^{M} kt_i \cdot \log^2(kt_i)\right)$.

Improving verification time. Our proofs for batch polynomial evaluation facilitate the reduction of the STIR verifier complexity by utilizing the prover to compute Ans_i and V_i much faster. Following the interaction phase, the prover derives the queries $\mathcal{Q}_i$ that the verifier will make to f_i and sends to the verifier the points $y_{i,x}$ and $z_{i,x}$ where (in the honest case), for every $x \in \mathcal{Q}_i$, it holds that $\mathsf{Ans}_i(x) = y_{i,x}$ and $V_i(x) = z_{i,x}$. The prover and verifier then engage in the batch polynomial evaluation protocol of Theorem 4.3 in order to prove that these are, indeed, the correct evaluations of the polynomials. Now that the verifier is convinced of the validity of these points, it can run the STIR verification procedure as if it had computed the evaluations independently.

Overall, the verifier performs $O(k \cdot \sum_{i \in [M]} t_i)$ field operations, reducing the factors of $\log^2(t_i + k \cdot t_{i+1})$. This comes at a minor cost to prover communication as the prover must send an additional $O(k \cdot \sum_{i \in [M]} t_i)$ field elements. We further note that this optimization is likely to be useful in the implementation of other protocols that utilize similar techniques, such as the ARC accumulation scheme [BMNW24].

Comparison with WHIR. This optimized protocol, which we refer to as STIR-SHAKE asymptotically matches the verifier complexity of the recently introduced WHIR [ACFY25] protocol. WHIR is also an interactive oracle proof of proximity for Reed–Solomon[4] which has a similar

[3] An interactive oracle proof of proximity is a generalization of an interactive proof, where the prover messages and the original input are oracles to which the verifier makes a small number of queries (rather than reading them in their entirety).

[4] WHIR also introduces the ability to verify a more general class of codes called "constrained" Reed–Solomon codes, but we focus on testing proximity to Reed–Solomon codes in this comparison.

basic structure to STIR-SHAKE, and where the verifier has the same asymptotic complexity of $O(\sum_{i=1}^{M} k \cdot t_i)$ field operations.

Acknowledgments

Gal Arnon is supported by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547), and Stellar Foundation Grant. Alessandro Chiesa and Giacomo Fenzi are supported in part by the Ethereum Foundation. Eylon Yogev is supported by the Israel Science Foundation (Grant No. 2302/22), European Research Union (ERC, CRYPTOPROOF, 101164375). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work was done in part while Gal Arnon and Giacomo Fenzi were visiting the Simons Institute for the Theory of Computing, and Giacomo Fenzi was visiting Succinct.

References

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "STIR: Reed–Solomon Proximity Testing with Fewer Queries". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 380–413.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "WHIR: Reed–Solomon Proximity Testing with Super-Fast Verification". In: *Proceedings of the 44th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '25. 2025.

[AFLN24] Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. "SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions". In: *Proceedings of the 43rd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '24. 2024, pp. 90–119.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments". In: *Proceedings of the 41st Annual International Cryptology*. CRYPTO '21. 2021, pp. 649–680.

[BDRNT25] Dan Boneh, Trisha Datta, Fernando Rex, Kamilla Nazirkhanova, and Alin Tomescu. "Dekart-Proof: Efficient Vector Range Proofs and Their Applications". In: *IACR Cryptol. ePrint Arch.* (2025), p. 1159.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 677–706.

[BFST03] C Sidney Burrus, James W Fox, Gary A Sitton, and Sven Treitel. "Horner's method for evaluating and deflating polynomials". In: *DSP Software Notes, Rice University, Nov* 26 (2003).

[BMNW24] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. "Arc: Accumulation for Reed-Solomon Codes". In: *IACR Cryptol. ePrint Arch.* (2024), p. 1731.

[But23] Vitalik Buterin. *Barycentric Evaluation Trees for Polynomial Commitments*. `https://hackmd.io/@vbuterin/barycentric_evaluation`. Accessed: 2025-05-21. 2023.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '23. 2023, pp. 499–530.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. "Polynomial Commitments from Lattices: Post-quantum Security, Fast Verification and Transparent Setup". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 207–242.

[CY24] Alessandro Chiesa and Eylon Yogev. "Building Cryptographic Proofs from Hash Functions". In: *URL: https://github. com/hash-based-snargs-book* (2024).

[FMN24] Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. "Lattice-Based Polynomial Commitments: Towards Asymptotic and Concrete Efficiency". In: *J. Cryptol.* 37.3 (2024), p. 31.

[Fid72] Charles M. Fiduccia. "Polynomial evaluation via the division algorithm the fast Fourier transform revisited". In: *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*. STOC '72. Denver, Colorado, USA, 1972, 88–93.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)* Cambridge University Press, 2013.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *Journal of the ACM* 62.4 (2015), 27:1–27:64.

[GR17] Oded Goldreich and Guy N. Rothblum. *Simple doubly-efficient interactive proof systems for locally-characterizable sets*. ECCC TR17-018. 2017.

[GR18] Oded Goldreich and Guy N. Rothblum. "Counting t-Cliques: Worst-Case to Average-Case Reductions and Direct Interactive Proof Systems". In: *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '18. 2018, pp. 77–88.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '10. 2010, pp. 177–194.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39.4 (1992), pp. 859–868.

[Mon87] Peter L Montgomery. "Speeding the Pollard and elliptic curve methods of factorization". In: *Mathematics of computation* 48.177 (1987), pp. 243–264.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. "Greyhound: Fast Polynomial Commitments from Lattices". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 243–275.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. "Signatures of Correct Computation". In: *Proceedings of the 10th Theory of Cryptography Conference*. TCC '13. 2013, pp. 222–242.

[RR20] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. FOCS '20. 2020, pp. 846–857.

[RR22] Noga Ron-Zewi and Ron D. Rothblum. "Proving as Fast as Computing: Succinct Arguments with Constant Prover Overhead". In: *Proceedings of the 54th ACM Symposium on the Theory of Computing*. STOC '22. 2022, pp. 1353–1363.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *Proceedings of the 48th ACM Symposium on the Theory of Computing*. STOC '16. 2016, pp. 49–62.

[Sha92] Adi Shamir. "IP = PSPACE". In: *Journal of the ACM* 39.4 (1992), pp. 869–877.

[Tre04] Lloyd N. Trefethen. "Barycentric Lagrange Interpolation". In: *SIAM Review* 46.3 (2004), pp. 501–517.