10

Interactive Oracle Proofs

10.1 IOPs: Definition and Associated Succinct Arguments

The concrete costs of the PCP prover of the previous section are very large. In this section, we describe more efficient protocols that operate in a generalization of the PCP setting, called Interactive Oracle Proofs (IOPs). Introduced by [44], [214], IOPs in fact generalize both PCPs and IPs. An IOP is an IP where, in each round the verifier is not forced to read the prover's entire message, but rather is given query access to it, meaning it can choose to look at any desired symbol of the message at the "cost" of a single query. This enables the IOP verifier to run in time sub-linear in the total proof length (i.e., the sum of the lengths of all the messages sent by the prover during the IOP).

Ben-Sasson *et al.* [44] showed that any IOP can be transformed into a non-interactive argument in the random oracle model using Merkle-hashing and the Fiat-Shamir transformation, in a manner entirely analogous to the Kilian-Micali transformation from PCPs to succinct arguments of Section 9.2. Specifically, rather than sending the IOP prover's message in each round of the IOP, the argument system prover sends a Merkle-commitment to the IOP prover's message. The argument system verifier then simulates the IOP verifier to determine which

elements of the message to query, and the argument system prover reveals the relevant symbols of the message by providing authentication paths in the Merkle tree. The interactive argument is then rendered non-interactive using the Fiat-Shamir transformation.[1]

The IOPs of This Section. In this section, we give an IOP for R1CS-satisfiability that is concretely much more efficient for the prover than the PCP of the previous section. The IOP can be understood as a combination of two constituent protocols. The first is a so-called *polynomial IOP* [83]; this is a variant of the IOP model described shortly. The specific polynomial IOP for R1CS that we cover, and optimizations thereof, was developed over a sequence of works [43], [98], [100].

The second is a *polynomial commitment scheme* (a notion introduced in Section 7.3) that is itself instantiated via an IOP. We give two such IOP-based polynomial commitment schemes in this section: one called FRI (short for Fast Reed-Solomon Interactive Oracle Proof of Proximity) with polylogarithmic proof length [34] (Section 10.4), and another implicit in a system called Ligero [7] with larger proofs but a concretely faster prover (Section 10.5). We also cover a generalization of Ligero with interesting performance characteristics, namely *asymptotically optimal* prover runtime, and no restrictions on the underlying field [67], [69], [137]. We refer to this as the Brakedown commitment, as that is the name of the first practical implementation of this variant [137].

10.2 Polynomial IOPs and Associated Succinct Arguments

As with standard IOPs introduced in Section 10.1 above, a polynomial IOP is an interactive proof, except that a subset of the prover's messages are not read in full by the verifier $\mathcal{V}$—let us call these messages "special". In a standard IOP, each special message is a string, and the verifier is given query access to individual symbols of the string. In a polynomial IOP, each special message i specifies a *polynomial* h_i over a finite field $\mathbb{F}$, with degree at most some specified upper bound d_i. In the IOPs of

[1] In the random oracle model, this IOP-to-SNARK transformation preserves both standard soundness and knowledge-soundness of the underlying IOP—see the end of Section 9.2.1 for details.

this section, h_i will always be a *univariate* polynomial, but in general h_i may be a multivariate polynomial.

Think of h_i as having a very large number of coefficients—in fact, in the polynomial IOP for R1CS given in this section, the degree d_i of h_i may be as large as the entire R1CS instance. This is why we do not want $\mathcal{V}$ to have to read a description of h_i in full, as that would require far more time than we'd like $\mathcal{V}$ to have to spend to check the proof. Rather, $\mathcal{V}$ is given query access to evaluations of h_i, meaning that $\mathcal{V}$ can choose any input r to h_i and learn $h_i(r)$.

Roughly speaking, the polynomial commitment schemes we cover in this section (Sections 10.4 and 10.5) allow the special messages themselves to be "implemented" via standard IOPs. That is, each polynomial h_i will be specified via a certain string m_i. An IOP will be given such that, when the verifier requests $h_i(r)$ and the prover sends back a claimed evaluation v_i, the verifier is able to confirm that m_i indeed specifies a polynomial h_i of the prescribed degree, with $v_i = h_i(r)$.

In summary, when one takes a polynomial IOP for R1CS-satisfiability, and replaces each "special message" and associated evaluation queries with a polynomial commitment scheme based on a standard IOP as above, the entire protocol is a standard IOP, which can then be transformed into a succinct argument via the transformation of [44].

Even if the polynomial commitment scheme is *not* implemented via a standard IOP (as with the schemes of Sections 14–16) one can still obtain a succinct argument via the following three-step design process.

- First, design a public-coin polynomial IOP for circuit- or R1CS-satisfiability.
- Obtain a public-coin, interactive succinct argument by replacing each "special" message h_i in the polynomial IOP with a polynomial commitment scheme.
- Remove interaction via Fiat-Shamir.

In fact, as explained next, all SNARKs covered in this survey are designed via this recipe, with the lone exception of those based on linear PCPs (Section 17).

Recasting Sections 7 and 8 as Polynomial IOPs. In Section 10.6, we recast the IP- and MIP-derived succinct arguments of Sections 7 and 8 as polynomial IOPs (in which there is a *single* special message, sent at the start of the protocol, which specifies a *multilinear* rather than univariate polynomial). This recasting provides a unified view of IP-, MIP-, and IOP-based SNARKs, and allows for a clean comparison of the pros and cons of the various approaches.

Relevant Costs of a Polynomial IOP. In a SNARK resulting from the above three-step design paradigm, the prover must (a) compute the polynomial h_i contained in each special message, and commit to it with the relevant polynomial commitment scheme (b) answer each evaluation query $h_i(r)$ made by the verifier and produce an associated evaluation-proof as per the polynomial commitment scheme (c) compute any non-special messages in the polynomial IOP. In practice, (a) and (b) are often the concrete bottlenecks for the prover. When h_i is a univariate polynomial, these costs grow at least linearly with the degree d_i of h_i, and hence a major goal when designing polynomial IOPs is to keep d_i linear in the size of the circuit or R1CS instance under consideration.

In terms of proof length and verifier time, verifying the evaluation-proofs sent in (b) above is often the concrete bottleneck. Hence, to minimize verification costs in the resulting SNARKs, a major goal is to minimize the number of evaluation queries that the polynomial IOP verifier makes to special messages.

10.3 A Polynomial IOP for R1CS-satisfiability

10.3.1 The Univariate Sum-Check Protocol

The key technical fact exploited in this section relates the sum of any low-degree polynomial over a potentially large subset of inputs H to the polynomial's evaluation at a *single* input, namely 0. Below, a non-empty subset $H \subseteq \mathbb{F}$ is said to be a multiplicative subgroup of field $\mathbb{F}$ if H is closed under multiplication and inverses, i.e., for any $a, b \in H$, $a \cdot b \in H$, and $a^{-1}, b^{-1} \in H$.

Fact 10.1. Let $\mathbb{F}$ be a finite field and suppose that H is a multiplicative subgroup of $\mathbb{F}$ of size n. Then for any polynomial q of degree less than $|H| = n$, $\sum_{a \in H} q(a) = q(0) \cdot |H|$. It follows that $\sum_{a \in H} q(a)$ is 0 if and only if $q(0) = 0$.

We provide a proof of this fact. Our proof assumes several basic results in group theory, and may be safely skipped with no loss of continuity.

Proof. When H is a multiplicative subgroup of order n, it follows from Lagrange's Theorem in group theory that $a^n = 1$ for any $a \in H$. Hence, H is precisely the set of n roots of the polynomial $X^n - 1$, i.e.,

$$\prod_{a \in H} (X - a) = X^n - 1. \tag{10.1}$$

We begin by proving the fact for $q(X) = X$, i.e., we show that $\sum_{a \in H} a = 0$. Indeed, it is easily seen that the coefficient of X^{n-1} when expanding out the left hand side of Equation (10.1) equals $-\sum_{a \in H} a$, and this must equal 0 because the coefficient of X^{n-1} on the right hand side of Equation (10.1) is 0.

Now let $q(X)$ be any monomial $X \mapsto X^m$ for $1 < m < n$. It is known that any multiplicative subgroup of a finite field $\mathbb{F}$ is cyclic, meaning there is some generator h such that $H = \{h, h^2, \ldots, h^n\}$. Then

$$\sum_{a \in H} q(a) = \sum_{a \in H} a^m = \sum_{j=1}^{n} h^{m \cdot j}. \tag{10.2}$$

Another application of Lagrange's theorem implies that if m and n are coprime, then h^m is also a generator of H, and hence $\sum_{j=1}^{n} h^{m \cdot j} = \sum_{j=1}^{n} h^j = \sum_{a \in H} a = 0$, where the final equality was established above.

If m and n are not coprime, then it is known that the order of h^m is $d \coloneqq \gcd(m, n)$, and hence letting $H' \coloneqq \{h^m, h^{2m}, \ldots, h^{(n/d)m}\}$, H is the disjoint union of the sets H', $h \cdot H'$, $h^2 \cdot H'$, $\ldots$, and $h^{d-1} \cdot H'$, where for any $a \in \mathbb{F}$, $a \cdot H'$ denotes the set $\{a \cdot b \colon b \in H'\}$.

Since H' is a multiplicative subgroup of order n/d, the reasoning in the first paragraph of the proof shows that $\sum_{a \in H'} a = 0$. Hence, the right hand size of Equation (10.2) equals $(1 + h + h^2 + \ldots + h^d) \cdot \sum_{a \in H'} a = 0$.

The lemma now follows for general polynomials $q(X) = \sum_{i=1}^{n-1} c_i X^i$ by linearity, combined with the fact that for any constant $c \in \mathbb{F}$, $\sum_{a \in H} c = |H| \cdot c$. □

For the remainder of this section, let H be a multiplicative subgroup of $\mathbb{F}$ of size n as in Fact 10.1. Let p be any univariate polynomial of degree at most D, where D may be greater than $|H| = n$. Recall that

$$\mathbb{Z}_H(X) = \prod_{a \in H} (X - a)$$

denotes the vanishing polynomial of H. Note that Equation (10.1) implies that $\mathbb{Z}_H(X) = X^n - 1$, i.e., $\mathbb{Z}_H$ is sparse, and hence can be evaluated at any desired input r in time $O(\log n)$ via repeated squaring. We derive the following simple consequence of Fact 10.1.

Lemma 10.2. $\sum_{a \in H} p(a) = 0$ if and only if there exists polynomials h^*, f with $\deg(h^*) \leq D - n$ and $\deg(f) < n - 1$ satisfying:

$$p(X) = h^*(X) \cdot \mathbb{Z}_H(X) + X \cdot f(X). \tag{10.3}$$

Proof. Suppose first that Equation (10.3) holds. Then clearly

$$\begin{aligned} \sum_{a \in H} p(a) &= \sum_{a \in H} (h^*(a) \cdot \mathbb{Z}_H(a) + a \cdot f(a)) = \sum_{a \in H} (h^*(a) \cdot 0 + a \cdot f(a)) \\ &= \sum_{a \in H} a \cdot f(a) = 0, \end{aligned}$$

where the final equality holds by Fact 10.1 and the fact that $X \cdot f(X)$ evaluates to 0 on input 0.

Conversely, suppose that $\sum_{a \in H} p(a) = 0$. Dividing p by $\mathbb{Z}_H$ allows us to write $p(X) = h^*(X) \cdot \mathbb{Z}_H(x) + r(X)$ for some remainder polynomial r of degree less than n. Since $0 = \sum_{a \in H} p(a) = \sum_{a \in H} r(a)$, we conclude by Fact 10.1 that r has no constant term. That is, we can write $r(X)$ as $X \cdot f(X)$ for some f of degree less than $n - 1$. □

The Univariate Sum-Check Protocol. Lemma 10.2 offers a polynomial IOP for verifiably computing sums of evaluations of univariate polynomials over multiplicative subgroup H (rather than sums of multivariate polynomial evaluations over the Boolean hypercube as considered

in the sum-check protocol of Section 4.2). Specifically, in order to prove that a specified univariate polynomial p of degree D sums to 0 over a multiplicative subgroup H with $|H| = n$, Lemma 10.2 implies that it is sufficient for a prover to establish that there exists functions h^* and f of degrees at most $D - n$ and $n - 1$ such that h^* and f satisfy Equation (10.3).

The natural way to accomplish this is to have the prover send two special messages specifying f and h^* respectively. The verifier can then confirm (with high probability) that Equation (10.3) holds by evaluating the left hand side and right hand side of Equation (10.3) at a random point $r \in \mathbb{F}$ and checking that the two evaluations are equal. This requires the verifier to evaluate p, f, and h^* at a single point r. Since both the right hand side and left hand side of Equation (10.3) are polynomials of degree at most $\max\{D, n\}$, up to soundness error $\max\{D, n\}/|\mathbb{F}|$ over the choice of r, if Equation (10.3) holds at the randomly chosen point r, then it is safe for the verifier to believe that Equation (10.3) holds as an equality of formal polynomials.

Remark 10.1. It is also possible to give an analogous IOP for confirming that p sums to 0 over an *additive* rather than multiplicative subgroup H of $\mathbb{F}$. This can be useful when working over fields of characteristic 2 (i.e., of size equal to a power of 2), since if a field has size 2^k for positive integer k, then it has an additive subgroup H of size $2^{k'}$ for every positive integer $k' < k$; moreover the vanishing polynomial $\mathbb{Z}_H(Y) = \prod_{a \in H}(a - h)$ is sparse (just as in the PCP of Section 9.4.1).

10.3.2 A Polynomial IOP for R1CS-SAT via Univariate Sum-Check

Motivation. In this section, we explain how to use the univariate sum-check protocol to give a polynomial IOP for R1CS-SAT. The reader may wonder, since IOPs are able to leverage interaction, why not just use the same techniques as in the MIP for R1CS-SAT of Section 8.4.2, which indeed we recast as a polynomial IOP in Section 10.6? The answer is that the MIP worked with multilinear polynomials over $O(\log n)$-variables, resulting in a protocol with at least $O(\log n)$ rounds. Here, we are seeking to have the prover only send univariate polynomials. This happens to

result in a polynomial IOP with just constantly many rounds.[2,3] As we discuss in Section 10.6, this ultimately leads to SNARKs with a different cost profile than MIP-derived SNARKs.

In summary, in this section we wish for the prover to exclusively send univariate polynomials, and hence we have to "redo" the MIPs of Section 8, replacing each constituent multilinear polynomial appearing in that protocol with a univariate analog.

Protocol Description. Recall from Section 8.4 that an R1CS-SAT instance is specified by $m \times n$ matrices A, B, C, and the prover wishes to demonstrate that it knows a vector z such that $Az \circ Bz = Cz$, where $\circ$ denotes Hadamard (entrywise) product. For simplicity, we assume that $m = n$ and that there is a multiplicative subgroup H of $\mathbb{F}$ of size exactly n. Let us label the n entries of z with elements of H, and let $\hat{z}$ be the unique univariate polynomial of degree at most $n-1$ over $\mathbb{F}$ that extends z in the sense that $\hat{z}(h) = z_h$ for all $h \in H$ (see Lemma 2.4). Similarly, let $z_A = Az$, $z_B = Bz$, and $z_C = Cz$ be vectors in $\mathbb{F}^n$, and let $\hat{z}_A$, $\hat{z}_B, \hat{z}_C$ extend z_A, z_B, z_C. To check that indeed $Az \circ Bz = Cz$, the verifier must confirm two properties. First:

$$\text{for all } h \in H,\ \hat{z}_A(h) \cdot \hat{z}_B(h) = \hat{z}_C(h). \tag{10.4}$$

Second:

$$\text{for all } h \in H, \text{ and } M \in \{A, B, C\},\ \hat{z}_M(h) = \sum_{j \in H} M_{h,j} \cdot \hat{z}(j). \tag{10.5}$$

Equation (10.5) ensures that z_A, z_B, z_C are indeed equal to Az, Bz, and Cz. Assuming this to be so, Equation (10.4) confirms that $Az \circ Bz = Cz$.

The prover sends four special messages, respectively specifying the degree-n polynomials $\hat{z}$, $\hat{z}_A$, $\hat{z}_B$, and $\hat{z}_C$.

[2] Of course, if the polynomial IOP is combined with an IOP-based polynomial commitment scheme such as FRI that uses logarithmically many rounds, then the resulting (standard) IOP will have logarithmically many rounds.

[3] Another benefit of having the prover send only univariate polynomials is that one of the two polynomial commitment schemes considered in this section, namely FRI, directly applies only to univariate polynomials. Though we nonetheless explain in Section 10.4.5 how to build upon such a polynomial commitment in an indirect manner to obtain one for multilinear polynomials in the IOP model, albeit with additional overheads.

Checking Equation (10.4). By Lemma 9.3 from the previous section, the first check is equivalent to the existence of a polynomial h^* of degree at most n such that

$$\hat{z}_A(X) \cdot \hat{z}_B(X) - \hat{z}_C(X) = h^*(X) \cdot \mathbb{Z}_H(X). \tag{10.6}$$

The prover sends a special message specifying the polynomial h^*. The verifier probabilistically checks that Equation (10.6) holds by choosing a random $r \in \mathbb{F}$ and confirming that

$$\hat{z}_A(r) \cdot \hat{z}_B(r) - \hat{z}_C(r) = h^*(r) \cdot \mathbb{Z}_H(r). \tag{10.7}$$

This requires querying the committed polynomials $\hat{z}_A$, $\hat{z}_B$, $\hat{z}_C$, and h^* at r; the verifier can evaluate $\mathbb{Z}_H(r)$ on its own in logarithmic time because $\mathbb{Z}_H(r)$ is sparse. Since all of the special messages sent by the prover are polynomials of degree at most n, up to soundness error $2n/|\mathbb{F}|$ over the choice of r, if Equation (10.7) holds at r then it is safe for the verifier to believe that Equation (10.6) holds, and hence also Equation (10.4).

Checking Equation (10.5). To check that Expression (10.5) holds, we leverage interaction, a resource that was not available to the PCP of Section 9.4. Fix $M \in \{A, B, C\}$ for the remainder of the paragraph. Let $\hat{M}(X, Y)$ denote the bivariate low-degree extension of the matrix M, interpreted in the natural manner as a a function $M(x, y)\colon H \times H \to \mathbb{F}$ via $M(x, y) = M_{x,y}$ That is, $\hat{M}(x, y)$ is the unique bivariate polynomial of degree at most n in each variable that extends M. Since $\hat{z}_M$ is the *unique* extension of z_M of degree less than n, it is easily seen that Equation (10.5) holds for all $h \in H$ if and only if the following equality holds as formal polynomials:

$$\hat{z}_M(X) = \sum_{j \in H} \hat{M}(X, j)\hat{z}(j). \tag{10.8}$$

Since any two distinct polynomials of degree at most n can agree on at most n inputs, if the verifier chooses r' at random from $\mathbb{F}$, then up to soundness error $n/|\mathbb{F}|$ over the choice of r', Equation (10.5) holds if and only if

$$\hat{z}_M(r') = \sum_{j \in H} \hat{M}(r', j)\hat{z}(j). \tag{10.9}$$

The verifier checks Equation (10.9) by sending r' to the prover and proceeding as follows. Let

$$q(Y) = \hat{M}(r', Y)\hat{z}(Y) - \hat{z}_M(r') \cdot |H|^{-1},$$

so that the validity of Equation (10.9) is equivalent to $\sum_{j \in H} q(Y) = 0$. The verifier requests that the prover establish that $\sum_{j \in H} q(Y) = 0$ by applying the univariate sum-check protocol from Section 10.3.1.

At the end of the univariate sum-check protocol applied to q, the verifier needs to evaluate q at a randomly chosen point r''. Clearly this can be done in a constant number of field operations if the verifier is given $\hat{z}(r'')$, $\hat{z}_M(r')$, and $\hat{M}(r', r'')$. The first two evaluations, $\hat{z}(r'')$ and $\hat{z}_M(r')$, can be obtained with one query each to the special messages specifying the polynomials $\hat{z}$ and $\hat{z}_M$.

How the Verifier Computes $\hat{M}(r', r'')$. All that remains is to explain how and when the verifier can efficiently obtain $\hat{M}(r', r'')$. For some "structured" matrices M, it is possible that $\hat{M}$ may be evaluatable in time polylogarithmic in n. This is analogous to how the verifier in the GKR protocol or the MIP of Section 8.2 avoids pre-processing so long as the multilinear extensions of the wiring predicates of the circuit- or R1CS-satisfiability instance can be evaluated efficiently.

For unstructured matrices M, time linear in the number K of nonzero entries of M may be required to evaluate $\hat{M}(r', r'')$ (Equation (10.11) later in this section offers one way $\hat{M}(r', r'')$ can be evaluated in this time bound). If one is unhappy with this runtime for the verifier, one can seek to have a *trusted* party, in pre-processing, *commit* to $\hat{M}$, which then permits the *untrusted* prover to efficiently and verifiably reveal $\hat{M}(r', r'')$ to the verifier as needed during the polynomial IOP just described.[4] Ideally, the pre-processing time, and the runtime of the prover when revealing $\hat{M}(r', r'')$ to the verifier, will be just linear in

[4] As observed in [230], one can reduce the work done by the trusted party by orders of magnitude, by having an untrusted party commit to the $\hat{M}$, and the trusted party to merely evaluate $\hat{M}$ at a random point. The trusted party then asks the untrusted party to reveal the committed polynomial's evaluation at that same point. If the two evaluations are equal, then (up to some negligible soundness error) it is safe to believe that the committed polynomial is $\hat{M}$.

the number K of nonzero entries of M. This goal is sometimes called *holography* [98] or *computation commitments* [226].

If the matrix M were *dense* (i.e., with $K = \Omega(n^2)$), it would be straightforward to use polynomial commitment schemes such as those given in this section (Section 10.4.2 and 10.5) or Section 14 to accomplish this goal. But typically R1CS matrices are *sparse*, meaning K is $\Theta(n)$ (see for example Section 8.4.1). In this case, the goal is more challenging to achieve.

Chiesa *et al.* [98], [100] nonetheless give a way to achieve it. Their technique is analogous in many ways to the commitment scheme for sparse *multilinear* polynomials described later in this survey (Section 16.2), which can be used to commit to the sparse multilinear extensions $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ of the wiring predicates used in the GKR protocol and the MIPs of Section 8, thereby achieving holography for those protocols. In both cases, the general idea is to express the "sparse" polynomial to be committed in terms of a constant number of *dense* polynomials, each of which can be straightforwardly committed in time linear in the sparsity K.[5]

Overview of Achieving Holography. The key to achieving this in the IOP setting is to give an explicit expression for $\hat{M}$, analogous to how Lemma 4.9 from Section 4.6.7.1 represents the *multilinear* extension of any function in terms of a higher-degree extension of the function. In more detail, recall that in Lemma 4.9, we defined $\widetilde{\beta}$ to be the unique multilinear extension of the "equality function" that takes two inputs from the Boolean hypercube and outputs 1 if and only if they are equal (see Equation (4.19)). In the setting of this section, the analog of the Boolean hypercube is the subgroup H, and the analog of $\widetilde{\beta}$ is the following bivariate polynomial: $u_H(X, Y) := \frac{\mathbb{Z}_H(X) - \mathbb{Z}_H(Y)}{X - Y}$. Though it is not immediately obvious, u_H is a polynomial of degree at most $|H| = n$

[5] Both here and in Section 16.2, K refers to the number of nonzero evaluations of the sparse polynomial over the relevant interpolating set defining the polynomial. In this section, that set is $H \times H$. In Section 16.2 and its application to the GKR protocols and the MIPs of Section 8, the relevant interpolating set is the Boolean hypercube.

in each variable.[6] For example, if $\mathbb{Z}_H(X) = X^n - 1$, then

$$u_H(X,Y) \coloneqq \frac{X^n - Y^n}{X - Y} = X^{n-1} + X^{n-2}Y + X^{n-3}Y^2 + X^{n-4}Y^3 + \ldots + XY^{n-2} + Y^{n-1}. \tag{10.10}$$

It is easy to check that for $x, y \in H$ with $x \neq y$, $u_H(x,y) = 0$. While less obvious, it is also true that for all $x \in H$, $u_H(x,x) \neq 0$ (though unlike $\widetilde{\beta}$, it is not necessarily the case that $u_H(x,x) = 1$ for all $x \in H$. For example, in Equation (10.10), $u_H(x,x) = nx^{n-1}$.).

Let $\mathcal{K}$ be a multiplicative subgroup of $\mathbb{F}$ of order K. Let us define three functions val, row, col mapping $\mathcal{K}$ to $\mathbb{F}$ as follows. We impose some canonical bijection between the nonzero entries of M and $\mathcal{K}$, and for $\kappa \in \mathcal{K}$, we define $row(\kappa)$ and $col(\kappa)$ to be the row index and column index of the κ'th nonzero entry of M, and define $val(\kappa)$ to be the value of this entry, divided by:

$$u_H(row(\kappa), row(\kappa)) \cdot u_H(col(\kappa), col(\kappa)).$$

Let $\hat{val}$, $\hat{row}$, and $\hat{col}$ be their unique extensions of degree at most K. Then we can express

$$\hat{M}(X,Y) = \sum_{\kappa \in \mathcal{K}} u_H(X, \hat{row}(\kappa)) \cdot u_H(Y, \hat{col}(\kappa)) \cdot \hat{val}(\kappa). \tag{10.11}$$

Indeed, it is easy to see that the right hand side of the above equation has degree at most $|H|$ in both X and Y, and agrees with $\hat{M}$ at all inputs in $H \times H$. Since $\hat{M}$ is the unique polynomial with these properties, the right hand side and left hand side are the same polynomial.

A First Attempt. Equation (10.11) expresses $\hat{M}$ in terms of degree-κ polynomials $\hat{row}$, $\hat{col}$, and $\hat{val}$, which suggests the following approach to permitting the verifier to efficiently learn $\hat{M}(r', r'')$ at the end of the polynomial IOP. The pre-processing phase can have a trusted party

[6] To see that $p_1(X,Y) \coloneqq \mathbb{Z}_H(X) - \mathbb{Z}_H(Y)$ is divisible by $p_2 \coloneqq X - Y$, observe that standard properties of polynomial division imply that when p_1 is divided by p_2, the remainder polynomial $r(X,Y)$ can be taken to have degree in X strictly less than that of p_2 in X, which is 1. Hence, $r(X,Y)$ has degree 0 in X. Since p_1 is symmetric, it can be seen that r is also symmetric, and hence $r(X,Y)$ is constant.

commit to the polynomials $\hat{val}$, $\hat{row}$, $\hat{col}$ (note that these are degree-K polynomials) and then when the verifier needs to know $\hat{M}(r', r'')$, the univariate sum-check protocol is invoked to establish that the polynomial

$$p(\kappa) \coloneqq u_H(r', \hat{row}(\kappa)) \cdot u_H(r'', \hat{col}(\kappa)) \cdot \hat{val}(\kappa) \tag{10.12}$$

sums to the claimed value over inputs in $\mathcal{K}$.

This unfortunately does not yield the efficiency we desire, because $u_H(r', \hat{row}(\kappa))$ and $u_H(r'', \hat{col}(\kappa))$ have degree as large as $n \cdot K$, since u_H has degree n in both of its variables. This means that applying the univariate sum-check protocol to $p(\kappa)$ would require the prover to send a polynomial h^* of degree $\Theta(n \cdot K)$, when we are seeking a prover runtime (and hence degree bound on all special messages) proportional just to K.

The Actual Holography Protocol. In the actual protocol, the pre-processing phase still commits to the three degree-K polynomials $\hat{val}$, $\hat{row}$, and $\hat{col}$.

To address the issue with the first attempt, we have to modify the "online phase" of the protocol, whereby the prover reveals to the verifier $\hat{M}(r', r'')$ to reduce its cost for the prover. Let us define f to be the unique polynomial of degree at most K that agrees with p (Equation (10.12)) at all inputs in $\mathcal{K}$. We are going to have the prover commit to f, and in order for the verifier to be able to check that f and p agree at all inputs in $\mathcal{K}$, we will need to identify a new expression (simpler than Equation (10.12)) that describes p's values at inputs in $\mathcal{K}$.

Specifically, observe that for any $a \in \mathcal{K}$,

$$u_H(r', \hat{row}(a)) = \frac{\mathbb{Z}_H(r') - \mathbb{Z}_H(\hat{row}(a))}{(r' - \hat{row}(a))} = \frac{\mathbb{Z}_H(r')}{(r' - \hat{row}(a))},$$

where the final equality uses the fact that $\hat{row}(a) \in H$. Similarly, for any $a \in \mathcal{K}$,

$$u_H(r'', \hat{col}(a)) = \frac{\mathbb{Z}_H(r'')}{(r'' - \hat{col}(a))}.$$

Hence, for any $a \in \mathcal{K}$,

$$p(a) = \frac{\mathbb{Z}_H(r')\mathbb{Z}_H(r'') \cdot \hat{val}(a)}{(r' - \hat{row}(a)) \cdot (r'' - \hat{col}(a))}. \tag{10.13}$$

This discussion leads to the following protocol enabling the verifier to efficiently learn $\hat{M}(r', r'')$ following a pre-processing phase during which a trusted party commits to the degree-K polynomials $\hat{row}$, $\hat{col}$, and $\hat{val}$. First, the prover commits to the degree-K polynomial f defined above, and the prover and verifier apply the univariate sum-check protocol to compute $\sum_{a \in K} f(a)$. Recall from Equation (10.11) that if f is as claimed, then this quantity equals $\hat{M}(r', r'')$.

Second, observe that for all $a \in \mathcal{K}$, $f(a)$ equals the expression in Equation (10.13) if and only if the following polynomial vanishes for all $a \in \mathcal{K}$:

$$(r' - \hat{row}(a)) \cdot (r'' - \hat{col}(a)) \cdot f - \mathbb{Z}_H(r')\mathbb{Z}_H(r'') \cdot \hat{val}(a). \qquad (10.14)$$

By Lemma 9.3, Expression (10.14) vanishes for all $a \in \mathcal{K}$ if and only if it is divisible by $\mathbb{Z}_{\mathcal{K}}(Y) = \prod_{a \in \mathcal{K}}(Y - a)$. The prover establishes this by committing to a polynomial q such that $q \cdot \mathbb{Z}_{\mathcal{K}}$ equals Expression (10.14), and the verifier checks the claimed polynomial equality by confirming that it holds at a random input $r''' \in \mathbb{F}$. This does require the verifier to evaluate $\hat{row}$, $\hat{col}$, $\hat{val}$, f, q, and $\mathbb{Z}_{\mathcal{K}}$ at r'''; the first three evaluations can be obtained from the pre-processing commitments to these polynomials, while $f(r''')$ and $q(r''')$ can be obtained from the prover's commitments to f and q, and $\mathbb{Z}_{\mathcal{K}}(r)$ can be computed in logarithmic time because it is sparse.

The polynomials that the prover commits to in the univariate sum-check protocol and in verifier's second check (namely, f and q) have degree at most $2K$.

Costs of the Polynomial IOP. Ignoring holography, the prover in the above polynomial IOP for R1CS-SAT sends five polynomials of degree at most n to check Equation (10.4): $\hat{z}$, $\hat{z}_A$, $\hat{z}_B$, $\hat{z}_C$, and h^*, each of which the verifier queries at a single point r. To check Equation (10.5) for each $M \in \{A, B, C\}$, the prover sends two polynomials of degree at most n as part of the sum-check protocol. During the univariate sum-check protocol, the verifier evaluates each of these polynomials at a random point r'' and also evaluates $\hat{z}$ at r'' and $\hat{z}_M$ at r'. In summary, if implemented naively, the prover in the polynomial IOP commits to 11 polynomials of degree at most n, and makes a total of 17 evaluation

queries to the various polynomials. The number of evaluation queries can be reduced to 12 as follows: one can use the same random values r' and r'' for all three instances of Equation 10.5; also, by performing all evaluation queries at the end of the protocol, it is safe for the verifier to set $r = r''$.

[43], [98], [100] describe a number of additional optimizations that improve concrete efficiency of the polynomial IOP and/or the resulting SNARK when the polynomial IOP is combined with various polynomial commitment schemes. We briefly describe one of these optimizations for illustration. When evaluating multiple different committed polynomials at the same point r, as the verifier in the above polynomial IOP does, it is typically more efficient, at least for the proof length and verifier time, to "batch-verify" the claimed evaluations, rather than perform each verification independently.[7] Exactly how much more efficiently depends on the polynomial commitment scheme used—the IOP-based polynomial commitments covered in this section have worse amortization properties than the homomorphic commitments of Section 14. Section 16.1 has details in the homomorphic case.

This type of efficient batch-verification of multiple evaluations of committed polynomials will recur in Section 18.

10.4 FRI and Associated Polynomial Commitments

10.4.1 Overview

FRI was introduced by Ben-Sasson *et al.* [34], and its analysis has been improved over a sequence of works [36], [49], [221]. While we defer details of how FRI works until Section 10.4.4, it is useful now to precisely state the guarantee that it provides. Let d be a specified degree bound. The prover's first message in FRI specifies a function $g\colon L_0 \to \mathbb{F}$, where L_0 is carefully chosen subset of $\mathbb{F}$. The prover claims that g is a polynomial of degree at most d; an equivalent way of saying this is that g is a *codeword* in the *Reed-Solomon code* of degree d. L_0 is chosen to have size $\rho^{-1} \cdot d$, where $0 < \rho < 1$ is a specified constant that is referred

[7] Batching techniques are also known when evaluating multiple different committed polynomials at distinct points rather than the same point [64].

to as the *rate* of the Reed-Solomon code that the FRI prover claims g is a codeword in. In practice, protocols that use FRI choose $\mathbb{F}$ to be significantly bigger than L_0, because the message size (and hence prover runtime) is lower-bounded by $|L_0|$ (hence $|L_0|$ should be kept as small as possible) yet $|\mathbb{F}|$ should be large to ensure a strong soundness guarantee.

The "remainder" of the FRI protocol is an IOP with the following guarantee. For specified parameter $\delta \in (0, 1 - \sqrt{\rho})$, known analyses of FRI guarantee that if the verifier accepts, then with overwhelming probability (say, at least $1 - 2^{-128}$), g is within relative distance δ of some polynomial p of degree at most d. That is, the number of points $r \in L_0$ for which $g(r) \neq p(r)$ is at most $\delta \cdot |L_0|$.

The query complexity of the FRI IOP is the dominant factor determining the proof length in succinct argument systems derived thereof, as each IOP query translates into a Merkle-tree authentication path that must be sent in the resulting argument system (see Section 9.2). Meanwhile, the prover runtime in FRI is mainly determined by the rate parameter ρ. This is because the smaller ρ is chosen, the longer the prover's messages in the IOP, and hence the bigger the prover runtime to generate those messages. However, we will see that smaller choices of ρ potentially permit the FRI verifier to make fewer queries for a given security level, and hence keeps the proof shorter when the IOP is ultimately converted into an argument system. Argument system designers can choose ρ to obtain their preferred tradeoff between prover time and proof size.

10.4.2 Polynomial Commitments and Queries to Points Outside of L_0

We highlight the following subtlety of FRI. As we have seen (e.g., Section 10.2), the prevailing paradigm for SNARK design demands the functionality of a polynomial commitment scheme. That is, the prover in the IOP must somehow send or commit to a low-degree polynomial p and the verifier must be able to force the prover to later evaluate the committed polynomial p at any point $r \in \mathbb{F}$ of the verifier's choosing.

A natural attempt to use FRI to achieve this functionality is the following. To commit to a degree d polynomial p, the prover would send a function g (claimed to equal p) by specifying g's values over L_0 (a strict subset of $\mathbb{F}$). And the verifier can run FRI to confirm that (with overwhelming probability) g has relative distance at most δ from some degree d polynomial p. Note that if $\delta < \frac{1-d/|L_0|}{2} = \frac{1-\rho}{2}$, then p is unique, i.e., there can be only one degree d polynomial within relative distance δ of g. This is because any two distinct polynomials of degree at most d can agree on at most d points (Fact 2.1).

Already, there is the nuisance that g is only guaranteed to be *close* to p, not exactly equal to p. This is closely analogous to the "relaxed" nature of the polynomial commitment scheme arising in Section 7 and the MIPs of Section 8.

But there is an additional issue as well: since g is only specified via its evaluations at inputs in L_0, how can the verifier determine evaluations of p on inputs $r \in \mathbb{F} \setminus L_0$? The research literature posits two approaches to dealing with this. The first is to carefully design polynomial IOPs, so that the verifier need not ever evaluate a polynomial specified by the prover at an input outside of L_0 (for brevity, we do not cover this approach in this survey). The second approach utilizes an observation that will recur later in this survey when we cover pairing-based polynomial commitment schemes (Section 15.2). Specifically, for any degree-d univariate polynomial p, the assertion "$p(r) = v$" is equivalent to the assertion that there exists a polynomial w of degree at most $d-1$ such that

$$p(X) - v = w(X) \cdot (X - r). \tag{10.15}$$

This is a special case of Lemma 9.3.

As observed in [240], the above observation means that in order to confirm that $p(r) = v$, the verifier can apply FRI to the function $X \mapsto (g(X) - v) \cdot (X - r)^{-1}$ using degree bound $d-1$ (we define this function to be 0 at input r). Note that whenever the FRI verifier queries this function at a point in L_0, the evaluation can be obtained with one query to g at the same point. If the FRI verifier accepts, then with overwhelming probability this function is within distance δ of some polynomial q of degree at most $d-1$. Since g and p have relative distance

at most δ over domain L_0, this means that the polynomials $q(X)(X-r)$ and $p(X)-v$ agree on at least $(1-2\delta)\cdot|L_0|$ inputs in L_0, and both have degree at most d.

Suppose that $\delta < \frac{1-\rho}{2}$, which guarantees that $(1-2\delta)|L_0| > d$. Since any two distinct polynomials of degree at most d can agree on at most d inputs, this implies that $q(X)\cdot(X-r)$ and $p(X)-v$ are the same polynomial, and this in turn implies by Equation (10.15) that $p(r)=v$.

In summary, if the prover sends a function $g\colon L_0 \to \mathbb{F}$ and convinces the FRI verifier that g has distance at most $\delta < \frac{1-\rho}{2}$ from some degree d polynomial p, and moreover the FRI verifier accepts when applied to $X \mapsto (g(X)-v)\cdot(X-r)^{-1}$ using degree bound $d-1$, then with overwhelming probability, $p(r)$ indeed equals v. That is, the verifier can safely accept that the low-degree polynomial p committed to via g evaluates to v at input r.

Note that the prover in this polynomial commitment scheme is bound to an actual polynomial p of degree at most d, in the sense that the prover must answer any evaluation request at input $r \in \mathbb{F}$ with $p(r)$ in order to convince the verifier to accept with non-negligible probability. This is in contrast to FRI by itself, which only binds the prover to a function g over domain L_0 that is *close to* p.

In summary, the technique of this section addressed *both* issues that prevented FRI from giving a polynomial commitment scheme directly. As described, the technique introduces a concrete overhead for the verifier of a factor close to two. To commit to a degree-d polynomial p, the prover sends (a Merkle-hash of) all evaluations of p over domain L_0, and FRI is applied to confirm that the function g actually sent is indeed close to some degree d polynomial. But then when the verifier queries the committed polynomial p at a point r outside of L_0, FRI has to be applied a second time, to confirm that the function $(g(X)-v)\cdot(X-r)^{-1}$ is (close to) some polynomial of degree at most $d-1$.

In fact, the first application of FRI can be omitted, thereby avoiding the overhead above. Indeed, since the second application of FRI guarantees that $(g(X)-v)\cdot(X-r)^{-1}$ has relative distance at most δ over L_0 from a polynomial $q(X)$ of degree $d-1$, it follows that the degree-d polynomial $p(X) \coloneqq q(X)\cdot(X-r)+v$ has relative distance at

most δ over L_0. Guaranteeing the existence of such a polynomial p was the entire point of the first application of FRI.

10.4.3 Costs of FRI

Prover Time. In applications of FRI (e.g., transforming the polynomial IOP of Section 10.3 into a SNARK), the prover will know the coefficients of a degree-d polynomial p or its evaluations on a size-d subset of L_0. To apply FRI to p, the prover must evaluate p at the remaining points in L_0. This is the dominant cost in terms of prover runtime. The fastest known algorithms for this are essentially FFTs, and they require $\Theta(|L_0| \cdot \log |L_0|) = \Theta(\rho^{-1} d \log(\rho^{-1} d))$ field operations. For constant rate parameters ρ, this is $\Theta(d \log d)$ time. We remark that Ben-Sasson *et al.* [34] describe the prover time in FRI as $O(d)$ field operations, but this assumes that the prover already knows the evaluations of p at all inputs in L_0, which will not be the case in applications of FRI.

Proof Length. FRI can be broken into two phases: a *commitment phase* and a *query* phase. The commitment phase is when the prover sends all of its messages in the IOP phase (during this phase, the verifier need not actually query any of the prover's messages) and the verifier sends one random challenge to the prover in each of $\log_2 |L_0|$ rounds.

The query phase is when the verifier actually queries the prover's messages at the necessary points to check the prover's claims. The query phase in turn consists of a "basic protocol" that must be repeated many times to ensure good soundness error. Specifically, in the basic query protocol the verifier makes 2 queries to each each of the $\log_2 |L_0|$ messages sent by the prover. To ensure a $2^{-\lambda}$ upper bound on the probability that the FRI verifier accepts a function g of relative distance more than δ from any degree-d polynomial, the basic protocol must be repeated roughly $\lambda / \log_2 (1/(1-\delta))$ times. This query phase is the dominant cost in the proof length of the argument systems obtained by combining FRI-based IOPs with Merkle-hashing. The argument system prover must send a Merkle-tree authentication path (consisting of $O(\log d)$ hash values) for each query in the IOP, the proof length of

the resulting arguments is $O\left(\lambda \cdot \log^2(d)/\log_2\left(1/\left(1-\delta\right)\right)\right)$ hash values. For constant values of δ, this is $O(\lambda \cdot \log^2(d))$ hash values.

Remark 10.2. The FRI proof system is largely independent of the setting of the parameter δ. The only reason that the prover and verifier within FRI need to "know" what δ will be set to in the soundness analysis is to ensure that they repeat the query phase of FRI at least $\lambda/\log_2(1/(1-\delta))$ times.

10.4.4 Details of FRI: Better Reed-Solomon Proximity Proofs via Interaction

Recall that the PCP for Reed-Solomon testing sketched in Section 9.4.3 worked by iteratively reducing the problem of testing a function G_i for proximity to a degree d_i polynomial to the problem of testing a related function G_{i+1} for proximity to a degree d_{i+1} polynomial where $d_{i+1} \ll d_i$ (more precisely, $d_{i+1} \approx \sqrt{d_i}$). A source of inefficiency in this construction was that each iterative reduction incurred a constant-factor loss in the distance from any low-degree polynomial of the function being analyzed. That is, if G_i is at least δ_i-far from every degree d_i polynomial, then G_{i+1} is only guaranteed to be at least (δ_i/c_0)-far from every polynomial of degree at most $\sqrt{d_i}$ for some universal constant $c_0 > 1$. This constant-factor loss in distance per iteration meant that we had to keep the number of iterations small if we wanted to maintain meaningful soundness guarantees. This in turn meant we needed to make sure that we made a *lot* of progress in reducing the degree parameter in each iteration. This is why we choose for d_{i+1} to be just $\sqrt{d_i}$—this ensured the d_i fell doubly-exponentially quickly in i, i.e., only $\Theta(\log\log d_0)$ iterations were required before the degree became 0, i.e., the function G_i became constant.

Unlike PCPs, IOPs such as FRI are allowed to be interactive, and FRI exploits interaction to ensure that the distance parameter δ_i does *not* fall by a constant factor in each round. This permits FRI to use exponentially more iterations—$\Theta(\log d_0)$ rather than $\Theta(\log\log d_0)$—while maintaining meaningful soundness guarantees, with corresponding efficiency benefits.

Recall from Section 10.4.1 that FRI proceeds in two phases, a *commitment phase* and a *query* phase. The commitment phase is when the prover sends all of its messages in the IOP (during this phase, the verifier need not actually query any of the prover's messages) and the verifier sends one random challenge to the prover in each round. The query phase is when the verifier queries the prover's messages at the necessary points to check the prover's claims.

Comparison of the IOP Commitment Phase to Section 9.4.3. For simplicity, let us suppose that in round i of the IOP, G_i is a function defined over a multiplicative subgroup L_i of $\mathbb{F}$, where $|L_i|$ is a power of 2, and the current degree bound d_i is also a power of 2. In round 0, G_0 is the evaluation table of the polynomial defined over domain L_0, for which we are testing proximity to univariate polynomials of degree d_0.

Recall that in Section 9.4.3, to show that G_i was a degree d_i polynomial, for any desired polynomial q_i of degree $\sqrt{d_i}$, it sufficed for the PCP to establish that $G_i(z) = Q_i(z, q_i(z))$ for some bivariate polynomial Q_i of degree at most $\sqrt{d_i}$ in each variable. When G_i indeed has degree at most d_i, the existence of such a polynomial Q_i was guaranteed by Lemma 9.6.

In the FRI IOP, $q_i(z)$ will simply be z^2 (since this choice of q_i does not depend on i, we omit the subscript i from q henceforth). When G_i indeed has degree at most d_i, Lemma 9.6 guarantees that there is a $Q_i(X, Y)$ of degree at most 1 in X and at most $d_i/2$ in Y such that $Q_i(z, z^2) = G_i(z)$. Under this setting of $q_i(z)$, this representation of G_i has an especially simple form. Let $P_{i,0}$ (respectively, $P_{i,1}$) consist of all monomials of G_i of even (respectively, odd) degree, but with all powers divided by two and then replaced by their integer floor. For example, if $G_i(z) = z^3 + 3z^2 + 2z + 1$, then $P_{i,0} = 3z + 1$ and $P_{i,1}(z) = z + 2$. When $q(z) = z^2$, Lemma 9.6 is simply observing that we can ensure that $G_i(z) = Q_i(z, z^2)$ by defining $Q_i(z, y) \coloneqq P_{i,0}(y) + z \cdot P_{i,1}(y)$.

In the PCP for Reed-Solomon testing of Section 9.4.3, $q(z)$ was chosen to be a polynomial of degree $\sqrt{d_i}$ such that the size of the image $q(L_i)$ was much smaller than $|L_i|$ itself (smaller by a factor of $\sqrt{d_i}$). Similarly, when L_i is a multiplicative subgroup of $\mathbb{F}$, the map $z \to z^2$ is

two-to-one on L_i,[8] so under our choice of $q(z) \coloneqq z^2$, if we define

$$L_{i+1} = q(L_i), \tag{10.16}$$

then $|L_{i+1}| = |L_i|/2$.

Complete Description of the IOP Commitment Phase. After the IOP prover commits to the polynomial G_i defined over domain L_i, the IOP verifier chooses a random value $x_i \in \mathbb{F}$ and requests that the prover send it the univariate polynomial

$$G_{i+1}(Y) \coloneqq Q_i(x_i, Y) = P_{i,0}(Y) + x_i \cdot P_{i,1}(Y), \tag{10.17}$$

defined over the domain L_{i+1} given in Equation (10.16).

This proceeds for rounds $i = 0, 1, \ldots, \log_2(d_0)$. Finally for $i^* = \log_2(d_0)$, $G_{i^*}(Y)$ is supposed to have degree 0 and hence be a constant function. In this round, the prover's message simply specifies the constant C, which the verifier interprets to specify that $G_{i^*}(Y) = C$.

Query Phase. The verifier $\mathcal{V}$ repeats the following ℓ times, for a parameter ℓ we set later. $\mathcal{V}$ picks an input $s_0 \in L_0$ at random, and for $i = 0, \ldots, i^* - 1$, $\mathcal{V}$ sets $s_{i+1} = q(s_i) = s_i^2$. The verifier then wishes to check that $G_{i+1}(s_{i+1})$ is consistent with Equation (10.17) at input s_{i+1}, i.e., that $G_{i+1}(s_{i+1})$ indeed equals $Q_i(x_i, s_{i+1})$. We now explain how this check can be performed with two queries to G_i.

Let $g(X) \coloneqq Q_i(X, s_{i+1})$ and observe that $g(X)$ is a linear function in X. Hence the entire function g can be inferred from its evaluations at two inputs.

Specifically, let $s_i' \neq s_i$ denote the other element of L_i satisfying $(s_i')^2 = s_{i+1}$. Since we have assumed that L_i is a multiplicative subgroup of even order, L_i contains -1 (see Footnote 8), and hence $s_i' = -s_i$. We know that $g(s_i) = Q_i(s_i, s_{i+1}) = G_i(s_i)$, while $g(s_i') = Q_i(s_i', s_{i+1}) =$

[8] To see this, recall from the proof of Fact 10.1 that any multiplicative subgroup H of a finite field $\mathbb{F}$ is cyclic, meaning there is a $h \in H$ such that $H = \{h, h^2, \ldots, h^{|H|}\}$, where $h^{|H|} = 1$. If $|H|$ is even, this means that $H' \coloneqq \{h^2, h^4, \ldots, h^{|H|}\}$ is also a multiplicative subgroup of $\mathbb{F}$, of order $|H|/2$, and H' consists of all perfect squares (also known as quadratic residues) in H. For each element h^{2i} in H', h^{2i} is the square of both h^i and $h^{i+|H|/2} = -h^i$.

$G_i(s'_i)$. And since g is linear, these two evaluations are enough to infer the entire linear function g, and thereby evaluate $g(x_i)$. More concretely, it holds that

$$g(X) = (X - s_i) \cdot (s'_i - s_i)^{-1} \cdot G_i(s'_i) + (X - s'_i) \cdot (s_i - s'_i)^{-1} \cdot G_i(s_i),$$

as this expression is a linear function of X that takes the appropriate values at $X = s_i$ and $X = s'_i$.

Accordingly, to check that $G_{i+1}(s_{i+1})$ indeed equals $Q_i(x_i, s_{i+1})$, the verifier queries G_i at s'_i and s_i, and checks that

$$G_{i+1}(s_{i+1}) = (x_i - s_i) \cdot (s'_i - s_i)^{-1} \cdot G_i(s'_i) + (x_i - s'_i) \cdot (s_i - s'_i)^{-1} \cdot G_i(s_i). \tag{10.18}$$

Completeness and Soundness. Completeness of the protocol holds by design: it is clear that if G_0 is indeed a univariate polynomial of degree at most d_0 over domain L_0 and sends the prescribed messages, then all of the verifier's checks will pass. Indeed, all of the consistency checks will pass, and G_{i^*} will indeed be a constant function.

The state-of-the-art soundness guarantee for FRI is stated in Theorem 10.4 below. Its proof is quite technical and is omitted from the survey, but we sketch the main ideas in detail.

Worst-Case to Average-Case Reductions for Reed-Solomon Codes. The key technical notion in the analysis of FRI is the following statement. Let $f_1, \ldots, f_\ell$ be a collection of ℓ functions on domain L_i, and suppose that at least one of f_j has relative distance at least δ from every polynomial of degree at most d_i over L_i. Then if $f \coloneqq \sum_{j=1}^{\ell} r_j f_j$ denotes a random linear combination of $f_1, \ldots, f_\ell$ (i.e., each r_j is chosen at random from $\mathbb{F}$), then with high probability over the random choices of $r_1, \ldots, r_\ell$, f also has distance at least δ from every polynomial of degree at most d_i over L_i. This statement is far from obvious, and to give a sense of why it is true, in Lemma 10.3 below we prove the following weaker statement that does *not* suffice to yield a tight analysis of FRI because it incurs a factor-of-2 loss in the distance parameter δ. Lemma 10.3 is due to Rothblum *et al.* [217]; our proof follows the presentation of Ames *et al.* [7, Proof of Case 1 of Lemma 4.2] almost verbatim.

Lemma 10.3. Let $f_1, \ldots, f_\ell$ be a collection of ℓ functions on domain L_i, and suppose that at least one of the functions, say f_{j^*}, has relative distance at least δ from every polynomial of degree at most d_i over L_i. If $f \coloneqq \sum_{j=1}^{\ell} r_j \cdot f_j$ denotes a random linear combination of $f_1, \ldots, f_\ell$, then with probability at least $1 - 1/|\mathbb{F}|$, f has distance at least $\delta/2$ from every polynomial of degree at most d_i over L_i.

Proof. Let V denote the span of $f_1, \ldots, f_\ell$, i.e., V is the set of all functions obtained by taking arbitrary linear combinations of $f_1, \ldots, f_\ell$. Observe that a random element of V can be written as $\alpha \cdot f_{j^*} + x$ where α is a random field element and x is distributed independently of α. We argue that conditioned on any choice of x, there can be at most one choice of α such that $\alpha \cdot f_{j^*} + x$ has relative distance at most $\delta/2$ from some polynomial of degree at most d_i. To see this, suppose by way of contradiction that $\alpha \cdot f_{j^*} + x$ has relative distance less than $\delta/2$ from some polynomial p of degree d_i and $\alpha' \cdot f_{j^*} + x$ has relative distance less than $\delta/2$ from some polynomial q of degree d_i, where $\alpha \neq \alpha'$. Then by the triangle inequality, $(\alpha - \alpha')f_{j^*}$ has relative distance less than $\delta/2 + \delta/2 = \delta$ from $p - q$. This contradicts the assumption that f_{j^*} has distance at least δ from every polynomial of degree at most d_i. □

A line of work [7], [36], [49], [217], [221] has improved Lemma 10.3 to avoid the factor-of-2 loss in the distance parameter δ. That is, rather than concluding that the random linear combination f has relative distance at most $\delta/2$ from every low-degree polynomial, these works show that f has relative distance at most δ from any low-degree polynomial. Two caveats are that these improvements do require δ to be "not too close to 1", and they also have worse failure probability than the $1/|\mathbb{F}|$ probability appearing in Lemma 10.3—see Theorem 10.4 for details on these caveats.

Detailed Soundness Analysis Sketch for FRI. The soundness analysis overview provided here below is merely a sketch, and we direct the interested reader to [49, Section 7] for a very readable presentation of the full details. Suppose that the function G_0 over domain L_0 has relative distance more than δ from every degree d_0 polynomial. We must show that for every prover strategy, with high probability the

prover fails at least one of the FRI verifier's consistency checks during the Query Phase of FRI.

For any fixed value of x_0 chosen by the verifier, Equation (10.18) specifies a function G_1 over L_1 such that if the prover sends G_1 in round 1 of the Commitment Phase of FRI, then G_1 will always pass the verifier's consistency check. Namely if for any $s_1 \in L_1$ we let $s_0, s_0' \in L_0$ denote the two square roots of s_1, then

$$G_1(s_1) = (x_0 - s_0) \cdot (s_0' - s_0)^{-1} \cdot G_0(s_0') + (x_0 - s_0') \cdot (s_0 - s_0')^{-1} \cdot G_0(s_0). \tag{10.19}$$

Note that G_1 depends on x_0; when we need to make this dependence explicit, we will write G_{1,x_0} rather than G_1.

In round 1 of the Commitment Phase of FRI, a prover can hope to "luck out" in one of two ways. The first way is if the verifier happens to select a value $x_0 \in \mathbb{F}$ such that G_{1,x_0} has relative distance significantly less than δ from a polynomial of degree d_1. The second way is that the prover could send a message $G_1' \neq G_1$ such that G_1' is much closer to a low-degree polynomial than is G_1, and hope the verifier doesn't "detect" the deviation from G_1 via its consistency checks.

It turns out that the second approach, of sending $G_1' \neq G_1$, never increases the probability that the prover passes the verifier's checks. Roughly speaking, this is because any "distance improvement" that the prover achieves by sending a function G_1' that deviates from G_1 is compensated for by an increased probability that the prover fails the verifier's consistency checks.

Let us now explain why the probability that the prover lucks out in the first sense is at most some small quantity ε_1. The idea is that G_{1,x_0} is essentially a random linear combination of $G_{1,0}$ and $G_{1,1}$. Specifically, since G_{1,x_0} is a linear function in x_0 (see Equation (10.19)), we can write $G_{1,x_0} = G_{1,0} + x_0 \cdot G_{1,1}$. Since x_0 is chosen by the verifier uniformly at random from $\mathbb{F}$ this means G_{1,x_0} is essentially a random linear combination of $G_{1,0}$ and $G_{1,1}$ (not quite, because the coefficient of $G_{1,0}$ is fixed to 1 rather than a random field element, but let us ignore this complication). Moreover, it is possible to show (though we omit the derivation) that if $G_{1,0}$ and $G_{1,1}$ are each of relative distance at most δ over L_1 from some polynomials $p(X)$ and $q(X)$ of degree less than

$d_1 = d_0/2$, then G_0 is of relative distance at most δ over L_0 from the polynomial $p(X^2) + X \cdot q(X^2)$, which has degree less than $2d_1 = d_0$, contradicting our assumption. The strengthening of Lemma 10.3 discussed earlier in this section asserted that a random linear combination of two functions, at least one of which has relative distance at least δ from every polynomial of degree d_1, is very likely to itself have relative distance at least δ from every such polynomial. Hence we reach the desired conclusion that with high probability over the choice of x_0, G_{1,x_0} has relative distance at least δ from every polynomial of degree at most d_1.

The above analysis applies for every round i, not just to round $i = 1$. Specifically, the optimal prover strategy sends a specified function $G_{i,x_{i-1}}$ in each round i, and with high probability every $G_{i,x_{i-1}}$ has relative distance at least δ from a polynomial of degree at most d_i. If this holds for the final round i^*, then in each repetition of the Query Phase of FRI, the verifier's final consistency check will reject with probability at least $1 - \delta$. This is because $G_{i^*,x_{i^*-1}}$ will have relative distance δ from a constant function, and the prover in the final round of the commitment phase is forced to send a constant C with the verifier checking in each execution of the query phase whether $G_{i^*,x_{i^*-1}}(s_{i^*}) = C$ for a point s_{i^*} that is uniformly distributed in L_{i^*}.

In summary, conditioned on the optimal prover strategy not "lucking out" during the commitment phase (which happens with probability at most ε_1 by the analysis sketched above), each repetition of the query phase will reveal an inconsistency with probability at least $1 - \delta$. So the probability that all ℓ repetitions of the query phase fail to detect an inconsistency is $\varepsilon_2 = (1 - \delta)^\ell$. Hence, if G_0 has relative distance more than δ from any polynomial of degree at most d_0, then the FRI verifier rejects with probability at least $1 - \varepsilon_1 - \varepsilon_2$. This is formalized in the following theorem from [36].

Theorem 10.4 ([36]). Let $\rho = d_0/|L_0|$, $\eta \in (0, \sqrt{\rho}/20)$, and $\delta \in (0, 1 - \sqrt{\rho} - \eta)$. If FRI is applied to a function G_0 that has relative distance more than δ from any polynomial of degree at most d_0, then the verifier accepts with probability at most $\varepsilon_1 + \varepsilon_2$, where $\varepsilon_2 = (1 - \delta)^\ell$ and $\varepsilon_1 = \frac{(d_0+1)^2}{(2\eta)^7 \cdot |\mathbb{F}|}$.

To give a sense of how the parameters in Theorem 10.4 may be set, [36] work through a numerical example, setting ρ to $1/16$, $|\mathbb{F}|$ to be 2^{256}, η to 2^{-14}, and δ to $1 - \sqrt{\rho} - \eta = 3/4 - \eta$. They show that if d_0 is at most $2^{16} = 65536$, then with $\ell := 65$ invocations of the basic query phase, this leads to soundness error $\varepsilon_1 + \varepsilon_2 \leq 2^{-128}$.

If using FRI-based polynomial commitments to transform the polynomial IOP of Section 10.3.2 into a holographic SNARK, this setting of d_0 is only sufficient to capture R1CS instances $Az \circ Bz = Cz$ such that each matrix A, B, C, has at at most $d_0/2 = 32768$ nonzero entries. Handling significantly larger R1CS instances at the same security level would require a larger field, and either more repetitions (increasing proof length and verification costs), or lower rate (further increasing prover time).

Theorem 10.4 incentivizes protocol designers to set δ as large as possible, because larger values of δ lead to smaller values of ε_2. However, in the context of using FRI-based polynomial commitments to transform polynomial IOPs into SNARKs, there may be other issues that prevent setting δ as large as it is set in the above numerical example, at least according to known soundness analyses. For example, deriving an actual polynomial commitment scheme from FRI as in Section 10.4.2 requires $\delta < (1 - \rho)/2 < 1/2$.[9] As another example, some concrete optimizations to various polynomial IOP in the literature that have been combined with FRI, e.g., [100, Theorem 8.2], require the yet more stringent condition $\delta \leq (1 - \rho)/3 < 1/3$. Under such a restriction, FRI would require over 200 repetitions of the query phase to achieve soundness error less than 2^{-128}.

It is conjectured that an analog of Theorem 10.4 holds even for δ as large as roughly $1 - \rho$ (see for example [35], [36]) rather than the $1 - \sqrt{\rho} - \eta$ bound on δ assumed in Theorem 10.4.

[9] RedShift [165] uses FRI to construct a relaxation of a polynomial commitment called a *list polynomial commitment*, and shows this relaxed primitive suffices for transforming some polynomial IOPs into SNARKs (though it drastically increases the cost of holography). This relaxed primitive can be achieved with δ as large as roughly $1 - \sqrt{\rho}$.

10.4.5 From Univariate to Multilinear Polynomials

FRI can be used to give a polynomial commitment scheme for *univariate* polynomials (Section 10.4.2). Zhang *et al.* [256] observe that, given any such commitment scheme for univariate polynomials, it is possible to devise one for multilinear polynomials in the following manner. As observed in Lemma 3.8, evaluating an ℓ-variate multilinear polynomial q over $\mathbb{F}$ at an input $r \in \mathbb{F}^\ell$ is equivalent to computing the inner product of the following two (2^ℓ)-dimensional vectors $u_1, u_2 \in \mathbb{F}^{2^\ell}$. Associating $\{0,1\}^\ell$ with $\{0, \ldots, 2^\ell - 1\}$ in the natural way, the w'th entry of the first vector u_1 is $q(w)$ and the w'th entry of the second vector u_2 is $\chi_w(r)$, the w'th Lagrange basis polynomial evaluated at r. This simple observation, that one can view the task of evaluating a polynomial p at an input r as an inner product between the coefficient vector u_1 of the polynomial over the relevant basis (in this case, the Lagrange basis), and the vector u_2 of all basis functions evaluated at r, will recur over and over again in the many polynomial commitment schemes discussed in this text.

Let H be a multiplicative subgroup of $\mathbb{F}$ of size $n := 2^\ell$. (A multiplicative subgroup of exactly this size only exists if and only if 2^ℓ divides $|\mathbb{F}| - 1$, so let us assume this). Let $b \colon H \to \{0,1\}^\ell$ be a canonical bijection. To commit to a multilinear polynomial q, it suffices to commit to a *univariate* polynomial Q over $\mathbb{F}$ of degree $|H| = n$ such that for all $a \in H$, $Q(a) = q(b(a))$, as q is fully specified by its evaluations over $\{0,1\}^\ell$. To later reveal $q(z)$ at a point $z \in \mathbb{F}^\ell$ of the verifier's choosing, consider the vector u_2 containing all Lagrange basis polynomials evaluated at z. It suffices to confirm that

$$\sum_{a \in H} Q(a) \cdot u_2(a) = v, \tag{10.20}$$

where v is the claimed value of $q(z)$ and here we associate the $|H|$-dimensional vector u_2 with a function over H in the natural way.

Let $\hat{u}_2$ be the unique polynomial of degree at most $|H|$ extending u_2, and let

$$g(X) = Q(X) \cdot \hat{u}_2(X) - v \cdot |H|^{-1}.$$

Observe that Equation (10.20) holds if and only if $\sum_{a \in H} g(a) = 0$. Hence, Equation (10.20) can be checked by applying the univariate

sum-check protocol described following Lemma 10.2 to the polynomial g. In more detail, in this protocol the prover sends a commitment to polynomials h^* and f such that

$$g(X) = h^*(X) \cdot \mathbb{Z}_H(X) + X \cdot f(X) \tag{10.21}$$

and f has degree at most $n - 1$. This requires the verifier to evaluate $g(r)$, $h^*(r)$, $\mathbb{Z}_H(r)$ and $f(r)$ for a randomly chosen $r \in \mathbb{F}$. As usual $\mathbb{Z}_H(r)$ is sparse so the verifier can compute this on its own in logarithmic time, and $h^*(r)$ and $f(r)$ can be obtained by querying the commitments to these polynomials.

Evaluating $g(r)$ requires evaluating $Q(r)$ and $\hat{u}_2(r)$. $Q(r)$ can be obtained by querying the commitment to Q. However, evaluating $\hat{u}_2(r)$ requires time linear in n (Lemma 3.8). Fortunately, the function $r \mapsto \hat{u}_2(r)$ is computed by a layered arithmetic circuit of size $O(n \log n)$, depth $O(\log n)$, and a wiring pattern for which $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ can be evaluated in $O(\log n)$ time for each layer i. Hence, the verifier can outsource the evaluation of $\hat{u}_2(r)$ to the prover using the GKR protocol (Section 4.6), with the verifier running in $O(\log^2 n)$ time.

Note that this transformation introduces considerable overhead. In order to commit to a single multilinear polynomial with n coefficients and later produce a single evaluation-proof, the prover has to commit to 3 univariate polynomials of degree n rather than just one such polynomial. Moreover, the prover and verifier have to apply the GKR protocol to a circuit of size superlinear in the number n of coefficients of the polynomial being committed.

10.5 Ligero and Brakedown Polynomial Commitments

In this section, we describe IOP-based polynomial commitment schemes with a much faster prover than FRI but larger evaluation proofs. For simplicity, we describe the polynomial commitment scheme here in the context of univariate polynomials (expressed over the standard monomial basis), but in fact the scheme applies directly to multilinear polynomials as well (see Figure 14.2 in Section 14 for details).

10.5.1 Identifying Tensor Product Structure in Polynomial Evaluation Queries

Let q be a degree-$(n-1)$ univariate polynomial over field $\mathbb{F}_p$ that the prover wishes to commit to, and let u denote the vector of coefficients of u. Then, as observed in the previous section, we can express evaluations of q as inner products of u with appropriate "evaluation vectors". Specifically, if $q(X) = \sum_{i=0}^{n-1} u_i X^i$, then for $z \in \mathbb{F}_p$, $q(z) = \langle u, y \rangle$ where $y = (1, z, z^2, \ldots, z^{n-1})$ consists of powers of z, and $\langle u, y \rangle = \sum_{i=0}^{n-1} u_i v_i$ denotes the inner product of u and y.

Moreover, the vector y has a tensor-product structure in the following sense. Let us assume that $n = m^2$ is a perfect square, and define $a, b \in \mathbb{F}^m$ as $a \coloneqq (1, z, z^2, \ldots, z^{m-1})$ and $b \coloneqq (1, z^m, z^{2m}, \ldots, z^{m(m-1)})$. If we view y as an $m \times m$ matrix with entries indexed as $(y_{1,1}, \ldots, y_{m,m})$, then y is simply the outer product $b \cdot a^T$ of a and b. That is, $y_{i,j} = z^{i \cdot m + j} = b_i \cdot a_j$ for all $0 \leq i, j \leq m-1$. Equivalently, we can write $q(z)$ as the vector-matrix-vector product $b^T \cdot u \cdot a$. This tensor structure is also exploited in several polynomial commitment schemes given in Section 14—Figure 14.1 in that section contains a pictorial example of this tensor structure.

10.5.2 Description of the Polynomial Commitment Scheme

Background on Error-Correcting Codes. To explain the commitment scheme, we need to introduce some terminology regarding error-correcting codes. An error-correcting code is specified by an encoding function E. E maps vectors in $\mathbb{F}^m$ to slightly longer vectors, in $\mathbb{F}^{\rho^{-1} \cdot m}$, where ρ is called the *rate* of the code (think of ρ as a constant such as $1/4$). E must be "distance-amplifying". This means that if messages $u_1, u_2 \in \mathbb{F}^m$ disagree in even a single coordinate, then $E(u_1)$ and $E(u_2)$ should disagree in a constant fraction of coordinates. The distance of the code is the minimum disagreement between any two codewords $E(u_1)$ and $E(u_2)$. The *relative distance* γ of the code is the distance divided by the codeword length.

The code is *linear* if E is a linear function. That is, $E(a \cdot u_1 + b \cdot u_2) = a \cdot E(u_1) + b \cdot E(u_2)$ for any messages $u_1, u_2 \in \mathbb{F}^m$ and scalars $a, b \in \mathbb{F}$.

A classic example of a linear code is the Reed-Solomon code. As we have seen throughout this survey, in this code a message $u_1 \in \mathbb{F}^m$ is interpreted as a degree-$m-1$ univariate polynomial p over $\mathbb{F}$. $E(u_1)$ is a list of $\rho^{-1}m$ evaluations of p. The distance of code is at least $(1-\rho)\cdot m$; this follows from the fact that any two distinct degree-$(m-1)$ polynomials can agree on at most $m-1$ inputs. For this code, $E(u_1)$ can be computed in time $O(\rho^{-1}m\log(\rho^{-1}m))$ using FFT-based multi-point evaluation algorithms.

For the rest of this section, let E denote the encoding function of a linear code with message length m, codeword length $\rho^{-1}m$, and constant relative distance $\gamma > 0$. Let us furthermore assume E is systematic, meaning for any message $u_1 \in \mathbb{F}^m$, the first m symbols of $E(u_1)$ are the entries of u_1 itself.

The commitment scheme described below is implicit in Ligero [7] in the case that E is the Reed-Solomon code (or, more precisely, its systematic variant, the univariate low-degree extension code, see Section 2.4). For general linear codes E, it is essentially implicit in work of Bootle *et al.* [67], see also [69]. Golovnev *et al.* [137] designed a concretely efficient error-correcting code E with a linear-time encoding procedure, and called the resulting implemented commitment scheme Brakedown (they also showed that the resulting polynomial commitment scheme is extractable). Xie *et al.* [253] refine the error-correcting code to improve performance, and use SNARK composition to reduce the length of the polynomial evaluation proofs in the resulting commitment scheme.

Commitment Phase. Recalling that $[m]$ denotes the set $\{1, 2, \ldots m\}$, let us view the coefficient vector u of q as an $m \times m$ matrix in the natural way (exactly as we viewed the vector y above as an $m \times m$ matrix). See Figure 14.1 in Section 14 for an example.

Let us denote the i'th row of u by u_i. Let $\hat{u}$ be the $m \times (\rho^{-1}m)$ matrix in which the i'th row is $E(u_i)$. In the IOP, the prover's commitment to u will simply be a message listing the entries of the matrix $\hat{u}$ (so in the final polynomial commitment scheme, the commitment to q will be the Merkle-hash of $\hat{u}$).

Let us denote the actual $m \times (\rho^{-1}m)$ matrix contained in the prover's commitment message by M. M is claimed to be $\hat{u}$, but if the prover

is cheating, then M may differ from $\hat{u}$. Upon receiving M, the verifier's initial goal is to try to ascertain whether or not M is actually a "well-formed" commitment matrix, meaning that every row of M is a codeword in the error-correcting code specified by E. The verifier will probabilistically check this via a "random linear combination of rows" test.

Specifically, the verifier chooses a random vector $r \in \mathbb{F}^m$ and sends r to the prover. The prover responds with a vector $w \in \mathbb{F}^{\rho^{-1}m}$ claimed to equal $r^T \cdot M$, and the verifier confirms that w is a codeword. More precisely, the verifier can confirm this by having the prover not send w itself, but instead send a message $v \in \mathbb{F}^m$, and the verifier sets w to $E(v)$. This means that the verifier reads v in its entirety.

For some integer parameter $t = \Theta(\lambda)$ that we will specify later, the verifier then randomly selects t entries of w and confirms that those entries are "consistent" with the actual commitment matrix M. That is, the verifier picks a size-t subset Q of the $\rho^{-1}m$ entries of w at random. For each $i \in Q$, the verifier "opens" *all* m entries in the ith column of M, and confirms that these entries are consistent with w_i, i.e., that $w_i = r^T \cdot M_i$ where M_i denotes the ith column of M. Since the verifier "consistency checks" t entries of w and each check requires opening an entire column of M, the total number of queries the verifier makes to entries of M is $t \cdot m$.

Evaluation Phase. Suppose the verifier requests that the prover reveal $q(z) = \langle u, y \rangle$ where u and y are defined as in Section 10.5.1. Recall that, viewing u as a matrix, $q(z) = b^T \cdot u \cdot a$, where $a, b \in \mathbb{F}^m$ are also as defined in Section 10.5.1. The evaluation phase is entirely analogous to the commitment phase, except that the random vector r used in the commitment phase is replaced with b.

In more detail, the prover first sends the verifier a vector v' claimed to equal $b^T \cdot u$, and analogous to the "random linear combination test", the verifier lets $w' := E(v')$. The verifier then picks a size-t subset Q' of columns of M and checks that for all $i \in Q'$, $w'_i = b^T \cdot M_i$, where M_i denotes the i'th column of M. If these checks all pass, then the verifier outputs $\langle w', a \rangle$ as its accepted value for $q(z)$.

This costs $t \cdot m$ queries to M. If the verifier's checks all pass, then the verifier outputs $\sum_{j=1}^{m} a_j \cdot v'_i$ as the evaluation $q(z)$ of the committed polynomial.

Intuition for Why This Scheme is Binding. If the prover is honest and every row of M is a codeword, then so will be any linear combination of the rows, by linearity of the code. Meanwhile, if M is "far" from having every row be a codeword (we make the relevant notion of "far" precise in the formal binding analysis), then it should be unlikely that the random linear combination $r^T M$ of the rows of M is close to any codeword z. In this event, since the verifier checks a large number of entries of z for consistency with the actual requested linear combination of the rows of M, the prover should fail one of the verifier's consistency checks with overwhelming probability.

So the "random linear combination" test roughly ensures that all rows of M are codewords, as claimed (this isn't quite true, but it is "close enough" to true for the remainder of the analysis to go through). If indeed all rows of M are codewords, let u_i be the message such that the i'th row of M equals $E(u_i)$, and let u denote the $m \times m$ matrix with i'th row equal to u_i. We claim that the prover is bound to answer any evaluation query $q(z)$ with $b^T \cdot ua$, meaning the prover is *bound* to the polynomial q whose coefficients are specified by the matrix u.

This holds because, by linearity of the code, the vector $b^T \cdot M$ requested in the evaluation phase is also a codeword. This means if the prover sends any codeword other than $b^T \cdot M$ in the evaluation phase, it will differ form $b^T \cdot M$ in many entries (by the distance properties of the code), and hence the verifier's consistency checks in that phase will detect the discrepancy with overwhelming probability. And because the code is systematic, if the prover indeed sends $b^T \cdot M$ in the evaluation phase, then the verifier outputs $b^T \cdot u \cdot a$ as the evaluation.

Details of Binding Analysis. For concreteness, we express the detailed binding analysis in the case that E is the Reed-Solomon encoding, but the analysis applies essentially unchanged to a general linear code E.

For the Reed-Solomon encoding of rate ρ, the relative distance of the code is greater than $1 - \rho$. This means that for any $\delta < (1 - \rho)/2$,

and for any vector $w \in \mathbb{F}^{\rho^{-1}m}$, there is at most one codeword within relative distance δ of w.

In this code, there is some designated set $L_0 \subseteq \mathbb{F}$ of size $\rho^{-1}m$, a message $u_i \in \mathbb{F}^m$ is interpreted as the evaluations of a degree-m polynomial p_i over the first m points in L_0, and the encoding of u_i is the vector of all evaluations of p_i at points in L_0.

First Attempt at a Detailed Binding Analysis. Lemma 10.3, which states that when taking a random linear combination of functions, if even a single one of the functions is far from all codewords in the Reed-Solomon code of a given degree, then (with probability at least $1 - 1/|\mathbb{F}|$) so is the random linear combination. Quantitatively, for any parameter $\delta > 0$, if any row of M has relative distance more than δ from every codeword, then Lemma 10.3 guarantees that, with probability at least $1 - 1/|\mathbb{F}|$, $r^T M$ has relative distance at least $\delta/2$ from every polynomial of degree at most m. In this event, since the verifier knows that z is a codeword, $r^T M$ and z differ in at least a $\delta/2$ fraction of their entries. Hence, the probability that $z_i = r^T \cdot M_i$ for all $i \in Q$ is at most $(1 - \delta/2)^t$. We can set t to ensure that this probability is below some desired soundness level, e.g., to ensure that $(1 - \delta/2)^t \leq 2^{-\lambda}$. In summary, we have established the following.

Claim 10.5. If the verifier's checks in the Commitment Phase all pass with probability more than $1/|\mathbb{F}| + (1 - \delta/2)^t$, then each row i of M has relative distance at most δ from some codeword.

Henceforth, let us assume that $\delta < (1 - \rho)/2$. This assumption combined with Claim 10.5 ensures that if the prover passes the verifier's checks with probability more than $(1 - \delta/2)^t$, then for each row M_i of M, there is a *unique* codeword p_i of degree at most m at relative distance at most δ from the ith row of M.

Unfortunately, the above is not enough on its own to guarantee that the scheme is binding. The reason for this is the following.

Claim 10.5 asserts that the verifier can be confident that each row i of M has relative Hamming distance at most δ from some codeword $p_i \in \mathbb{F}^{|L_0|}$. Let $E_i = \{j \colon p_{i,j} \neq M_{i,j}\}$ denote the subset of entries on which p_i and row i of M differ, and let $E = \cup_{i=1}^m E_i$. That is, E is the

set of columns such that at least one row i deviates from its closest polynomial p_i in that column. Claim 10.5 doesn't rule out the possibility $|E| = \sum_{i=1}^{m} |E_i|$. In other words, it leaves open the possibility that any two different rows i, i' of M deviate from the corresponding codewords $p_i, p_{i'}$ in different locations.

Full Binding Analysis. We need a refinement of Claim 10.5 that does rule out this possibility (this refinement also improves the the $\delta/2$ appearing in Claim 10.5 to δ). We do not prove this refinement—the interested reader can find the proof in [7, Lemma 4.2].

Claim 10.6. (Ames *et al.* [7, Lemma 4.2]) Suppose that $\delta < \frac{1-\rho}{3}$ and that the verifier's checks in the Commitment Phase all pass with probability more than $\varepsilon_1 := |L_0|/|\mathbb{F}| + (1-\delta)^t$. Let $E = \cup_{i=1}^{m} E_i$ be defined as above. Then $|E| \leq \delta \cdot |L_0|$.

To argue binding, let $h' := \sum_{i=1}^{m} a_i \cdot p_i$. We claim that, if the prover passes the verifier's checks in the Commitment Phase with probability more than $\varepsilon_1 = |L_0|/|\mathbb{F}| + (1-\delta)^t$ and sends a codeword h in the Evaluation Phase such that $h \neq h'$, then the prover will pass the verifier's checks in the Evaluation Phase with probability at most $\varepsilon_2 := (\delta + \rho)^t$. To see this, observe that h and h' are two distinct codewords in the Reed-Solomon code with message length m, and hence they can agree on at most m inputs. Denote this agreement set by A. The verifier rejects in the Evaluation Phase if there is any $j \in Q'$ such that $j \notin A \cup E$, where E is as in Claim 10.6. $|A \cup E| \leq |A| + |E| \leq m + \delta \cdot |L_0|$, and hence a randomly chosen column j of M is in $A \cup E$ with probability at most $m/|L_0| + \delta \leq \rho + \delta$. It follows that the verifier will reject with probability at least $1 - (\rho + \delta)^t$.

In summary, we have shown that for $\delta < (1-\rho)/2$, if the prover passes the verifier's checks in the Commitment Phase with probability at least $|L_0|/|\mathbb{F}| + (1-\delta)^t$, then the prover is *bound* to the polynomial q^* whose coefficient matrix u has i'th row equal to the first m symbols of p_i. That is, on any evaluation query z, the verifier either outputs $q^*(z)$, or else rejects in the Evaluation Phase with probability at least $1 - (\rho + \delta)^t$.

10.5.3 Discussion of Costs

Let λ denote a security parameter defined as follows. Suppose we wish to guarantee that if the prover convinces the verifier not to reject in either the Commitment Phase or Query Phase with probability at least $\varepsilon_1 + \varepsilon_2 = 2^{-\lambda}$, then the prover is forced to answer any evaluation query consistent with a fixed polynomial q^* of degree at most $n-1$. The costs of the polynomial commitment scheme are then as follows. Throughout, we suppress dependence on ρ^{-1} and δ, as we consider these parameters to be constants in $(0,1)$.

Commitment and Evaluation Proof Sizes. After some optimizations to the above commitment scheme (omitted from this survey for brevity), one can achieve a commitment that is just a single Merkle-hash, with evaluation proofs of size $O(\sqrt{n\lambda})$. The square root dependence on n arises because the vectors $v = r^T M$ and $v' = b^T \cdot u$ that the prover sends in response to the random linear combination test and evaluation query equals the number of columns of the matrices u and M, and because the verifier queries all entries of t columns of M, each of which has length equal to the number of rows of M. The matrix was chosen to have $\sqrt{n}$ rows and columns to balance these costs.

Prover Time. The prover's runtime in the argument resulting from this IOP is dominated by two operations. The first is to encode each row of the matrix M. If the Reed-Solomon code is used as in Ligero, this requires one FFT operation per row, on vectors of length $\Theta(\sqrt{n})$. Since each such FFT requires $O(\sqrt{n}\log n)$ field operations, the total runtime for the FFTs is $O(n \log n)$ field operations.

If an error-correcting code with linear-time encoding is used (e.g., the one designed in [137]), then the encoding operations require only $O(n)$ time in total.

The second bottleneck is the need for the prover to compute a Merkle-hash of its first message, which has length $O(n)$. This requires $O(n)$ cryptographic hash evaluations. In practice, it is typically the encoding operations that dominate the prover's runtime, not the Merkle-hashing.

Verifier Time. After some concrete optimizations, the verifier's runtime in the argument system is dominated by the need to apply the error-correcting code's encoding procedure to two vectors v and v' of length $O(\sqrt{n\lambda})$. If using the Reed-Solomon code, this is $O(\sqrt{n\lambda}\log n)$ field operations; if using a linear-time encodable code, this is $O(\sqrt{n\lambda})$ field operations.

Comparison to FRI. The communication complexity and verifier runtime of the argument system are much larger than those of FRI, at least asymptotically. Whereas the FRI proof length and verifier time are polylogarithmic in n, these costs for the Ligero- and Brakedown-polynomial commitments are proportional to the square root of n.

The main benefit of Ligero- and Brakedown- polynomial commitments is a significantly faster prover—over an order of magnitude in practice for large enough values of n [137]. In the case of Brakedown, the prover time is asymptotically optimal $O(n)$ instead of $O(n \log n)$. Additionally, Brakedown works over any field of sufficient size, whereas FRI and Ligero's commitment require a field $\mathbb{F}$ that supports FFTs, meaning $\mathbb{F}$ must have a multiplicative or additive subgroup of appropriate size.[10]

Ligero++ [53] combines Ligero's commitment and FRI to obtain a polynomial commitment scheme with similar prover time to Ligero's commitment, and similar proof length to FRI's. However, this approach increases the verifier's runtime to close to linear in n.

10.6 Unifying IPs, MIPs, and IOPs via Polynomial IOPs

Sections 7–10 described succinct arguments for circuit- and R1CS-satisfiability obtained by combining an IP or MIP with a polynomial commitment scheme. We described the protocol of this section (Section 10.3) as a polynomial IOP. In this section, we recast the IPs and MIPs in the same framework. This permits shorter descriptions of the protocols and clarifies the pros and cons of the various approaches. To

[10] Recent work [37], [38] provides FFT-like algorithms and associated argument systems running in $O(n \log n)$ time in arbitrary fields, but a more expensive field-dependent pre-processing phase is required and at the time of writing the concrete costs of these algorithms are unclear.

avoid redundancy, our descriptions here are somewhat sketchy, as this section merely recasts protocols described in full detail earlier in this monograph.

Polynomial IOP from the GKR Protocol. Here, we recast the IP-based succinct argument from Section 7 as a polynomial IOP. In this argument, the prover claims to know a witness $w \in \mathbb{F}^n$ such that $\mathcal{C}(w) = 1$, where $\mathcal{C}$ is a circuit known to both the prover and verifier. The one and only "special" message in this polynomial IOP is the first one. Specifically, the prover begins the protocol by sending the $(\log n)$-variate polynomial $\tilde{w}$, i.e., the multilinear extension of the witness w. The prover and verifier then apply the GKR protocol (Section 4.6) to the claim that $\mathcal{C}(w) = 1$. Note that the verifier in the GKR protocol does not need to know *anything* about w to execute its part of the protocol, until the very end of the protocol when it needs to know $\tilde{w}(r)$ for a randomly chosen $r \in \mathbb{F}^{\log n}$. This is why the verifier in this polynomial IOP does not need to learn the special message in full, but rather just a single evaluation of the polynomial $\tilde{w}$ described therein.

Polynomial IOP from Clover. Here, we recast the MIP-based succinct argument from Section 8.2 as a polynomial IOP (the MIP for R1CS from Section 8.4 can be similarly recast). The section defined the notion of a *correct transcript* W for the claim that $\mathcal{C}(w) = 1$: a transcript W assigns a value to each of the S gates of $\mathcal{C}$, and W is *correct* if it assigns the output gate value 1, and actually corresponds to the gate-by-gate evaluation of $\mathcal{C}$ on some input w.

As with the GKR-based polynomial IOP, the one and only "special" message in this polynomial IOP is the first one. The prover begins the protocol by sending the $(\log S)$-variate polynomial $\tilde{W}$, i.e., the multilinear extension of the correct transcript W.

Equation (8.2) in Section 8.2 defines a *derived polynomial* $h_{\tilde{W}}$ such that the following two properties hold: (1) $\tilde{W}$ extends a correct transcript

if and only if[11] $\sum_{u \in \{0,1\}^{3 \log S}} h_{\tilde{W}}(u) = 0$. (2) For any $r_1, r_2, r_3 \in \mathbb{F}^{\log S}$, $h_{\tilde{W}}(r_1, r_2, r_3)$ can be efficiently computed given $\tilde{W}(r_1)$, $\tilde{W}(r_2)$, $\tilde{W}(r_3)$.

Hence, the polynomial IOP simply applies the sum-check protocol to the polynomial $h_{\tilde{W}}$. Note that the verifier in the sum-check protocol does not need to know *anything* about $h_{\tilde{W}}$ to execute its part of the protocol, until the very end of the protocol when it needs to know $h_{\tilde{W}}(r_1, r_2, r_3)$ for a randomly chosen $(r_1, r_2, r_3) \in \mathbb{F}^{3 \log S}$. By the second property above, this evaluation can be efficiently obtained given $\tilde{W}(r_1)$, $\tilde{W}(r_2)$, $\tilde{W}(r_3)$.

Using the interactive proof of Section 4.5.2, the verifier can avoid making three evaluation-queries to $\tilde{W}$, instead making only a single evaluation query. Specifically, the verifier asks the prover to tell it (claimed values for) $\tilde{W}(r_1)$, $\tilde{W}(r_2)$, $\tilde{W}(r_3)$, and the technique of Section 4.5.2 allows the verifier to check this claim by evaluating $\tilde{W}(r_4)$ at a single randomly chosen point $r_4 \in \mathbb{F}^{\log S}$.

Remark 10.3. The above two protocol descriptions assume that the multilinear extensions of the "wiring predicates" of $\mathcal{C}$ used internally in the protocols can be evaluated quickly by the verifier. If not, holography can nonetheless be achieved via the commitment scheme for sparse multilinear polynomials given later, in Section 16.2.

Comparison of IP-, MIP-, and Constant-Round-Polynomial-IOP-Derived Arguments. The above recasting makes clear that in the arguments derived from IPs and MIPs above (which are both based on multilinear polynomials), the prover commits to only a *single* multilinear polynomial. In contrast, in the polynomial IOP of this section (Section 10.3), which exclusively uses univariate polynomials, the prover needs to commit to many polynomials, each at least as big as the polynomial committed from the IP- and MIP-derived arguments. Specifically, the prover from this section commits to 11 univariate polynomials, at least if naively implemented (and not counting the cost of holography). This is a major reason that arguments derived from constant-round

[11] More precisely, the definition of $h_{\tilde{W}}$ is randomized, with some small probability that $\tilde{W}$ does not extend a correct transcript, yet $h_{\tilde{W}}$'s evaluations over the Boolean hypercube do not sum to 0.

polynomial IOPs tend to be much more expensive for the prover in terms of both time and space requirements.[12, 13]

On the other hand, the fact that the univariate-polynomial-based IOP of this section is only a constant number of rounds is a significant benefit. This means that, if combined with a polynomial commitment scheme with constant-sized commitments and evaluation proofs (i.e., KZG commitments covered later in Section 15.2), it yields a SNARK with constant proof size.[14] In contrast, the use of multilinear polynomials and the sum-check protocol in the IP- and MIP-derived SNARKs results in at least logarithmically many rounds, and hence at least logarithmic proof size and verifier time.

In summary, the IP- and MIP-derived SNARKs tend to have much lower prover costs, but have higher verification costs than alternatives based on constant-round polynomial IOPs. The IP-based argument takes this to an extreme, because it applies the polynomial commitment scheme only to the circuit witness w; if w is smaller than the full circuit $\mathcal{C}$, this keeps prover costs low. But the resulting proofs can be quite large. Indeed, ignoring the cost of evaluation-proofs from the chosen polynomial scheme, the MIP-derived argument for circuit-satisfiability above has proofs that are shorter than the IP-based one by a factor roughly equal to the circuit depth. On the other hand, it applies the polynomial commitment scheme to the entire circuit transcript extension $\tilde{W}$ rather than just the witness extension $\tilde{w}$, which can lead to larger prover costs than the IP-derived arguments.

[12] The constant-round polynomial IOP described in this section underlies systems including Marlin [98] and Fractal [100]. Another popular constant-round polynomial IOP with a similar cost profile to Marlin is PlonK [123].

[13] There may be additional reasons constant-round polynomial IOPs are more expensive for the prover. For example, existing SNARKs derived from constant-round polynomial IOPs require the prover to perform FFTs, even when using a polynomial commitment scheme that does not require FFTs. This is not the case for the IP- and MIP-derived arguments above, which entirely avoid FFTs if they use a multilinear polynomial commitment scheme that does not require them.

[14] By constant proof size, we mean a constant number of elements of a cryptographic group $\mathbb{G}$.