IOPs with Inverse Polynomial Soundness Error

Gal Arnon
gal.arnon@weizmann.ac.il
Weizmann Institute

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

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

Abstract

We show that every language in NP has an Interactive Oracle Proof (IOP) with inverse polynomial soundness error and small query complexity. This achieves parameters that surpass all previously known PCPs and IOPs. Specifically, we construct an IOP with perfect completeness, soundness error $1/n$, round complexity $O(\log \log n)$, proof length $\mathrm{poly}(n)$ over an alphabet of size $O(n)$, and query complexity $O(\log \log n)$. This is a step forward in the quest to establish the sliding-scale conjecture for IOPs (which would additionally require query complexity $O(1)$).

Our main technical contribution is a *high-soundness small-query* proximity test for the Reed–Solomon code. We construct an IOP of proximity for Reed–Solomon codes, over a field $\mathbb{F}$ with evaluation domain L and degree d, with perfect completeness, soundness error (roughly) $\max\{1-\delta, O(\rho^{1/4})\}$ for δ-far functions, round complexity $O(\log \log d)$, proof length $O(|L|/\rho)$ over $\mathbb{F}$, and query complexity $O(\log \log d)$; here $\rho = (d+1)/|L|$ is the code rate. En route, we obtain a new high-soundness proximity test for bivariate Reed–Muller codes.

The IOP for NP is then obtained via a high-soundness reduction from NP to Reed–Solomon proximity testing with rate $\rho = 1/\mathrm{poly}(n)$ and distance $\delta = 1 - 1/\mathrm{poly}(n)$ (and applying our proximity test). Our constructions are direct and efficient, and hold the potential for practical realizations that would improve the state-of-the-art in real-world applications of IOPs.

Keywords: interactive oracle proofs; Reed–Solomon proximity testing; sliding-scale conjecture

Contents

1 Introduction

Probabilistic proofs are a central tool in complexity theory and cryptography. They have enabled breakthroughs in areas such as zero knowledge, delegation of computation, and hardness of approximation.

A probabilistically checkable proof (PCP) [BFLS91; FGLSS96] is a proof system in which a polynomial-time probabilistic verifier has query access to a proof string. The celebrated PCP theorem [AS98; ALMSS98] states that every language in NP has a PCP with constant soundness error, polynomial proof length, and constant query complexity. While decades of research have contributed numerous PCP constructions achieving important goals, major open problems remain. For example, the shortest PCPs known to date have quasilinear proof length [BS08; Din07], and achieving PCPs with linear proof length remains an open problem.

Some limitations of known PCPs have been circumvented by the introduction of a multi-round variant of PCPs called *interactive oracle proofs* (IOPs) [BCS16; RRR16]. An IOP is an interactive proof where the verifier has PCP-like access to each prover message (the verifier may read a few symbols from each prover message).

IOPs leverage interaction to overcome barriers that arise with PCPs. For instance, known PCPs use proof composition to reduce query complexity but at the cost of a blowup in proof length. In contrast, *interactive* proof composition enables IOPs to reduce query complexity with only a small increase in proof length. Thanks to this and many other techniques, known IOPs achieve linear proof length as well as other desirable properties such as fast provers, zero knowledge, and concrete efficiency [BCGV16; Ben+17; BCGRS17; BBHR18; BCGGHJ17; XZZPS19; BCG20; BCL22; RR20; ACY22a; ACY22b; BN22; RR22; GLSTW23]. IOPs now underlie highly-efficient cryptographic proofs that have seen widespread deployment in real-world applications.

Small soundness error. A major open problem for PCPs is the *sliding-scale conjecture*: for every $\beta \geq \frac{1}{\text{poly}(n)}$, every language in NP has a PCP with soundness error β, proof length $\text{poly}(n)$ over an alphabet of size $\text{poly}(1/\beta)$, and query complexity $O(1)$ [BGLR93]. Note that the PCP theorem does not achieve the requirement of small soundness error (e.g., take $\beta = 1/n$). This requirement is crucial for numerous applications; see [Mos19] for a survey on this conjecture and its implications.

The state-of-the-art for low-error PCPs is due to Dinur, Harsha, and Kindler [DHK15]. Building on a line of work on low-error PCPs [RS97; AS03; DFKRS11] and through comprehensive understanding and usage of proof composition, they show that every language in NP has a PCP with perfect completeness, soundness error $1/\text{poly}(n)$, proof length $\text{poly}(n)$ over an alphabet of size $\text{poly}(n)$, and query complexity $\text{polyloglog}(n)$.

Despite the striking progress on IOP constructions in the last few years, the aforementioned conjecture remains open even for IOPs and, in fact, *all known IOPs to date have not made any improvements compared to PCPs as far as soundness error is concerned.*[1] Indeed, leveraging interaction to achieve small soundness error has been a frustratingly elusive goal, and the aforementioned PCP from [DHK15] remains the state-of-the-art even for low-error IOPs.

Conjecture 1.1 (sliding-scale conjecture for IOPs)**.** *For every $\beta \geq \frac{1}{\text{poly}(n)}$, every language in* NP *has an IOP with perfect completeness, soundness error β, total proof length* $\text{poly}(n)$ *over an alphabet of size* $\text{poly}(1/\beta)$*, and query complexity $O(1)$; the round complexity can be up to* $\text{poly}(n)$*.*

[1] Any IOP can be "unrolled" into a corresponding PCP with the same verifier parameters. (Only the prover is affected: proof length, and thus prover time, blows up, and zero knowledge may disappear.)

The sliding-scale conjecture for IOPs is a natural generalization of the sliding-scale conjecture for PCPs. Progress on the IOP conjecture may lead to progress on the PCP conjecture. In fact, the PCP theorem was achieved thanks to advancements in IPs; and, relevant to our setting, [ABCY22] shows that solving the sliding-scale conjecture for IOPs with round complexity $\text{polylog}(n)$ implies solving the sliding-scale conjecture for PCPs. Finally, trading a larger alphabet for fewer queries has cryptographic applications: in constructions of succinct arguments from IOPs based on the Merkle-tree paradigm, the main bottleneck to reducing argument size comes from the query complexity of the IOP rather than the alphabet size or proof length of the IOP [BCS16; CY21a; CY21b].

1.1 Our results

We demonstrate that *interaction can improve soundness*, achieving a regime of parameters that is beyond all known PCPs and IOPs to date. The theorem below is a step towards proving the sliding scale conjecture for IOPs; in particular, fixing $\beta = 1/n$, we obtain that every language in NP has an IOP with soundness error $1/n$ and query complexity $O(\log\log n)$.

Theorem 1. *For every $\beta = \beta(n) \in (0,1)$, every language in* NP *has a public-coin IOP with perfect completeness, soundness error β, round complexity $O(\log\log n)$, proof length $\text{poly}(\frac{n}{\beta})$ over an alphabet of size $\text{poly}(\frac{n}{\beta})$, and query complexity $O(\log\log n)$ ($O(1)$ queries per round).*

Setting $\beta = 1/n$, Theorem 1 achieves the same soundness error as in [DHK15] with smaller query complexity (albeit at the price of more rounds of interaction). Our techniques differ significantly and are more "direct", in the sense that our protocol avoids proof composition and soundness amplification (see Section 1.2 for further discussion). Moreover, our IOP implies a corresponding low-error PCP: the IOP can be "unrolled" into a PCP with the same query complexity and soundness error, and proof length $n^{O(\log\log n)}$ (see [ABCY22]).

While Theorem 1 applies for general NP languages, for the NP-complete language $\mathsf{R1CS}$ we achieve better parameters. Over a field of size $O(n/\beta)$ we achieve proof length $O(t \cdot n \cdot \beta^{-1/t})$ and query complexity $O(t \cdot \log\log n)$ for any $t \in \mathbb{N}$. By setting t appropriately, we achieve short proof length. For example, for $\beta = 1/n$, by setting $t = \log\log\log n$ we get a high-soundness IOP for $\mathsf{R1CS}$ with proof length $n^{1+o(1)}$ and query complexity $\tilde{O}(\log\log n)$, which is essentially tight [ABCY22].[2]

Proximity testing for Reed–Solomon codes. The main technical tool underlying Theorem 1 is a new proximity test for Reed–Solomon (RS) codes. RS codes are a fundamental object of study in algebraic coding theory and theoretical computer science, and, in particular, they often play a role in the design of probabilistic proofs. The code $\mathsf{RS}[\mathbb{F}, L, d]$ is the set of functions $f\colon L \to \mathbb{F}$ that are evaluations of polynomials of degree (at most) d. An IOP of proximity (IOPP) for $\mathsf{RS}[\mathbb{F}, L, d]$ enables a prover to convince a verifier with oracle access to f that f is close (in relative Hamming distance) to a codeword in $\mathsf{RS}[\mathbb{F}, L, d]$ by engaging in a multi-round interaction where the prover sends additional oracles and the verifier sends randomness. We provide an IOPP for $\mathsf{RS}[\mathbb{F}, L, d]$ with small soundness error and small query complexity.

Theorem 2 (informal)**.** *Let $\mathbb{F}$ be a "nice" field. There is a public-coin IOPP for $\mathsf{RS}[\mathbb{F}, L, d]$ with soundness error $\max\{1-\delta,\ \rho^{1/4} + O(\frac{d^2 \cdot (1/\rho)^4}{|\mathbb{F}|})\}$ for δ-far functions, round complexity $O(\log\log d)$, proof length $O(|L|/\rho)$ over $\mathbb{F}$, and query complexity $O(\log\log d)$; here $\rho := \frac{d+1}{|L|}$ is the code rate.*

[2] Following [ABCY22], an IOP with the same parameters but shorter proof length $\tilde{O}(n)$ would imply a randomized algorithm that decides $\mathsf{R1CS}$ in subexponential time (in the witness length), opposing current beliefs of its hardness.

A formal statement and proof appear in Section 6.

The new proximity test is direct and efficient, and holds the potential for practical realizations that would improve the state-of-the-art in real-world applications of IOPs. Prior IOPPs for Reed–Solomon codes achieve, regardless of code rate, only constant soundness error or logarithmic query complexity (e.g., [Din07; BS08; Mie09; RVW13; BBHR18; BGHSV06; BGKS20; BCIKS20]).

A building block of independent interest on the way to Theorem 2 is a new proximity test for (individual degree) bivariate Reed–Muller codes.[3] Our bivariate proximity test has small soundness error and small query complexity, making it potentially useful in other contexts.

Theorem 3 (informal)**.** *Let $\mathbb{F}$ be a "nice" field. There is a public-coin IOPP for the bivariate Reed–Muller code $\mathsf{RM}[\mathbb{F}, X \times Y, (d_X, d_Y)]$ with soundness error $\max\left\{2\sqrt{1-\delta},\ \rho_X^{1/4}\right\} + O\left(\frac{d^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right)$ for δ-far functions, round complexity $O(\log\log d)$, proof length $O\left(d \cdot |X|/\rho_X\right)$ over $\mathbb{F}$, and query complexity $O(\log\log d)$; here $d := \max\{d_X, d_Y\}$ and $\rho_X := \frac{d_X+1}{|X|}$.*

A formal statement and proof appear in Section 8.3.

1.2 Related work

PCPs with small soundness error. The PCP Theorem [AS98; ALMSS98] and parallel repetition [Raz95] together imply a PCP with soundness error β, proof length $n^{O(\log 1/\beta)}$, and query complexity 2. The PCP in [MR08], with the amplification in [DS14], achieves soundness error β with only two queries, but the alphabet size is exponential in $1/\beta$. The constructions in [AS03; RS97; MR10; DFKRS11] achieve proof length $\mathrm{poly}(n)$ and query complexity $O(1)$, with soundness error $2^{-(\log n)^\alpha}$ for a constant $0 < \alpha < 1$.

The PCP in [DHK15] gets much closer to the sliding scale conjecture: it achieves soundness error $1/\mathrm{poly}(n)$, proof length $\mathrm{poly}(n)$ over an alphabet of size $\mathrm{poly}(n)$, and query complexity $\mathrm{polyloglog}(n)$. The PCP is the result of naively repeating, for $\mathrm{polyloglog}(n)$ many times, a PCP verifier with soundness error $n^{-1/\,\mathrm{polyloglog}(n)}$ and query complexity $\mathrm{polyloglog}(n)$. In particular, reducing the query complexity of their main construction to $O(1)$ would *not* resolve the sliding scale conjecture, as the soundness amplification would still result in $\mathrm{polyloglog}(n)$ queries. Our IOP avoids soundness amplification and directly achieves soundness error $1/\mathrm{poly}(n)$ and only $O(\log\log n)$ queries, with the potential for further optimizations.

IOPs. The last few years have seen tremendous progress on IOPs. Known IOP constructions demonstrate that even small round complexity (typically $O(1)$ or $O(\log n)$) suffices to achieve small (linear) proof length and fast (quasilinear or linear) proving time. Yet prior IOP constructions achieve only constant soundness error (and smaller soundness error is then achieved via repetition).

The sliding-scale conjecture for IOPs can serve as a stepping stone towards the sliding scale conjecture for PCPs [ABCY22]: any IOP with soundness error $1/\mathrm{poly}(n)$, round complexity $\mathrm{polylog}(n)$, proof length $\mathrm{poly}(n)$, and query complexity $O(1)$ can be transformed into a PCP with proof length $\mathrm{poly}(n)$ and (roughly) the same soundness error and query complexity. At present it is unknown whether higher round complexity (more than $\mathrm{polylog}(n)$) enables a similar implication from a low-error IOP to a low-error PCP.

Applying the transformation in [ABCY22] to Theorem 1, we obtain a PCP with soundness error $1/n$, proof length $n^{O(\log\log n)}$ over an alphabet of size $\mathrm{poly}(n)$, and query complexity $O(\log\log n)$.

[3] The bivariate Reed–Muller code $\mathsf{RM}[\mathbb{F}, X \times Y, (d_X, d_Y)]$ is the set of bivariate functions $f\colon X \times Y \to \mathbb{F}$ that are evaluations of polynomials with degree at most d_X in the first variable and at most d_Y in the second variable.

Proximity tests for RS codes. Our proximity test for Reed–Solomon codes improves over prior work in several ways.

The popular FRI protocol and its variants achieve a soundness error that tends to zero as the code rate tends to zero [BBHR18; BGKS20; BCIKS20]. However, to achieve even a constant soundness error, the FRI protocol requires round complexity $\Omega(\log d)$ and query complexity $\Omega(\log d)$. This is because the protocol works (in simple terms) by reducing the problem of testing degree d to the problem of testing degree $d/2$ using one round of interaction.

The Ben-Sasson–Sudan PCPP for Reed–Solomon codes [BS08] also follows a recursive structure but each recursive step reduces the degree d to $\sqrt{d}$. While this leads to only $O(\operatorname{loglog} d)$ recursive steps, the soundness error is $1-\frac{1}{\operatorname{polylog}(d)}$ even for functions that are far from the code. To compensate for this, soundness is amplified by repetition, yielding query complexity $\operatorname{polylog}(d)$.

Our proximity test overcomes both of these issues, using only $O(\operatorname{loglog} d)$ recursive steps and query complexity $O(\operatorname{loglog} d)$ while maintaining high soundness throughout. The proof length in our protocol is slightly higher at $O(|L|/\rho)$, compared to FRI, which has proof length $O(|L|)$, and [BS08], which has proof length $\widetilde{O}(|L|)$.

2 Techniques

We provide a high-level overview of our main result: an IOP for NP with small soundness error and small query complexity.

Our starting point is a polynomial IOP (poly-IOP) for the NP-complete language R1CS with constant query complexity and soundness error $1/\text{poly}(n)$. A poly-IOP [CHMMVW20; BFS20] is an IOP where the (honest and malicious) prover sends as its messages univariate polynomials of prescribed degrees over a certain finite field; the verifier has oracle access to these polynomials in the sense that it may query the evaluation of any polynomial at any point in the field.[4] High-soundness small-query poly-IOPs for NP are known (e.g., [BCRSVW19]), and any poly-IOP with these properties can be used to achieve Theorem 1. See Section 7 for the poly-IOP that we use.

Our goal is compile the given poly-IOP to a (standard) IOP while preserving high soundness and increasing the query complexity as little as possible. We achieve this in two steps. First, we show a high-soundness reduction from poly-IOP to testing proximity to Reed–Solomon codes. Then we show a high-soundness proximity test for Reed–Solomon codes.

(1) High-soundness reduction. We reduce the problem of verifying the given poly-IOP to testing whether a function f is a codeword in a Reed–Solomon code with degree $d := O(n)$ and rate ρ. In the completeness case, f is a valid codeword; in the soundness case, f is δ-far from the code for a large (relative) distance $\delta = \delta(\rho)$. Crucially, our reduction ensures that if $\rho = 1/\text{poly}(n)$ then $\delta = 1 - 1/\text{poly}(n)$. Prior reductions only achieve constant distance δ regardless of the code rate ρ; this precludes achieving inverse-polynomial soundness error with small query complexity q because all queries made by a verifier testing f might, with probability $O(2^{-\mathsf{q}})$, fall inside a set where f agrees with a low-degree polynomial. Thus, to achieve soundness error $1/\text{poly}(n)$ the proximity test of f must have query complexity $\mathsf{q} = \Omega(\log n)$ which is much larger than we want.

(2) High-soundness proximity test. We construct a proximity test for Reed–Solomon codes for degree d with query complexity $O(\text{loglog}\, d)$ such that when the tested function f is at distance δ from a Reed–Solomon code with rate ρ, the test accepts with probability at most (roughly) $\max\{1-\delta,\ O(\rho^{1/4})\}$. Thus if $d = O(n), \rho = 1/\text{poly}(n), \delta = 1 - 1/\text{poly}(n)$ (as in the high-soundness reduction) then query complexity is $O(\text{loglog}\, n)$ and the verifier accepts with probability at most $1/\text{poly}(n)$.

Unlike all prior proximity tests for RS codes, we design our proximity test *in the poly-IOP model*. More precisely, we design a polynomial IOP of proximity (poly-IOPP) for Reed–Solomon codes of degree d where the prover's messages are polynomials of degree at most $O(\sqrt{d})$. We then apply the high-soundness reduction described previously to reduce verifying the poly-IOPP into testing that a function f Reed–Solomon codeword *with degree* $O(\sqrt{d})$. Thus, we have reduced the problem of proving whether a function is of degree d to checking whether a related function is of degree $O(\sqrt{d})$.

This progress allows us to recursively apply the reduction $O(\text{loglog}\, d)$ times until the degree is constant, in which case the function's degree can be directly checked.

This approach raises two main challenges.

- The entire reduction from testing degree d to testing $\sqrt{d}$ must preserve distance with low error.

[4] Polynomial IOPs are typically used in a different context to ours, in combination with cryptographic polynomial commitment schemes in order to construct succinct arguments. Polynomial IOPs are similar but distinct from *RS-encoded IOPs* (where the prover's messages are RS codewords over a specific domain rather than over the entire field), which have been used in past IOP constructions (e.g., [RRR16; BCRSVW19] and others).

- We can afford to test only a *single* function of degree $\sqrt{d}$ as testing even two functions would result, via the recursion, in query complexity $\Omega(\log d)$. The high-soundness reduction must therefore reduce to testing a single function.

Organization. In Section 2.1, we outline how we compile a poly-IOP into an IOP using proximity testing for Reed–Solomon codes. In Section 2.2, we describe our poly-IOP for Reed–Solomon codes. Finally, in Section 2.3, we sketch the proof of Theorem 2, showing how to combine the tools developed in Section 2.1 and Section 2.2 to construct a proximity test for Reed–Solomon codes.

2.1 From poly-IOPP to IOPP

We outline how to reduce testing the validity of a poly-IOP where the prover's messages have degree at most d to the problem of testing that a single univariate function evaluated over some domain L is close to degree d. Crucially, if the prover is dishonest, then the resulting univariate function is $(1-\rho^{1/\Omega(1)})$-far from degree d over L, where $\rho = (d+1)/|L|$. In other words, we show how to compile a poly-IOP into a standard IOP, using a proximity test for Reed–Solomon codes. If the proximity test has high soundness for codewords that are far from low-degree, then the resulting IOP has high soundness.

Our transformation from a poly-IOP to a (standard) IOP is generic: it works for any poly-IOP for a relation R even if the prover messages have different degrees. Moreover, while in this overview we discuss only poly-IOPs, the transformation works also to transform a poly-IOPP into an IOPP. This flexibility allows us to use this transformation to construct our protocol for NP and also to design proximity tests in the poly-IOPP model.

Lemma 1 (informal)**.** *Suppose we have these ingredients:*

- *A poly-IOP for a relation R with perfect completeness, soundness error β, round complexity k, and query complexity q, where the prover's i-th message is a polynomial of degree at most d_i.*[5]
- *A proximity test for $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ for $d := \max_{i\in[\mathsf{k}]}\{d_i\}$ with perfect completeness, soundness error $\beta_{\mathsf{prx}}(\delta)$ for δ-far functions, round complexity $\mathsf{k}_{\mathsf{prx}}$, proof length $\mathsf{l}_{\mathsf{prx}}$, input query complexity $\mathsf{q}_{\mathsf{prx},f}$, and proof query complexity $\mathsf{q}_{\mathsf{prx},\pi}$.*

Then there is an IOP for R with perfect completeness, soundness error $\max\{\beta,\ \beta_{\mathsf{prx}}(1-\rho^{1/\Omega(1)}) + \mathrm{poly}(d, 1/\rho)/|\mathbb{F}|\}$*, round complexity $O(\mathsf{k}+\mathsf{k}_{\mathsf{prx}})$, proof length $O(\mathsf{k}\cdot|L| + \mathsf{l}_{\mathsf{prx}})$, and query complexity $O(\mathsf{q} + \mathsf{k}\cdot\mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\pi})$. Here $\rho := (d+1)/|L|$ is the rate of the code $\mathcal{C}$.*

We describe our compiler, starting with a naive construction.

Naive attempt. For every $i \in [\mathsf{k}]$, denote by $\hat{f}_i \in \mathbb{F}^{\le d_i}[X]$ the polynomial of degree at most d_i sent by the prover in round k. In order to move from the poly-IOP model to the IOP model, we need to have the prover send proof strings rather than polynomials. Hence, in round i, the prover sends $f_i \colon L \to \mathbb{F}$, the *evaluation* of $\hat{f}_i$ over the proximity test domain L; that is, $f_i \in \mathsf{RS}[\mathbb{F}, L, d_i]$ is such that $f_i(L) = \hat{f}_i(L)$. The verifier acts as in the original poly-IOP, where it queries f_i in place of $\hat{f}_i$. Finally, the verifier runs the RS proximity test on f_i in order to ensure that f_i is close to an RS codeword.

This reduction is insufficient for us for two main reasons.

[5] Our full theorem considers also poly-IOPs where the prover sends multiple polynomials in a single round.

1. If the prover sends f_i that is at a constant distance from an RS codeword, we cannot hope to simultaneously have high soundness and small query complexity, since all of the queries of the proximity test may land in a low-degree part of the function with too large a probability.

2. In the poly-IOP model the verifier can query each $\hat{f}_i$ at any point in the field $\mathbb{F}$. In the current approach the verifier has access to $\hat{f}_i(L)$, which implies that we need $L = \mathbb{F}$. This results in a large proof length, and may be incompatible with the low-degree test. While we could design the poly-IOP to be aware of the evaluation domain L, this complicates its construction and breaks the abstraction of a simple and convenient poly-IOP model.

We get around these issues using *quotienting* and *out of domain sampling.*

Enforcing consistency via univariate function quotienting. The *quotient* of a function f relative to $p\colon S \to \mathbb{F}$ with $S \subseteq \mathbb{F}$ is defined as:

$$\mathsf{Quotient}(f, S, p)(x) := \frac{f(x) - \hat{p}(x)}{\hat{V}_S(x)} \ ,$$

where $\hat{V}_S(X) := \prod_{a \in S}(X - a)$ is the vanishing polynomial over S, and $\hat{p}$ is the unique polynomial of degree $\leq |S| - 1$ such that $\hat{p}(a) = p(a)$ for every $a \in S$.

For a univariate polynomial $\hat{f}(X)$, if $\hat{f}(a) = p(a)$ for every $a \in S$, then $\hat{f}(X)$ can be written as $\hat{V}_S(X) \cdot \hat{g}(X) + \hat{p}(X)$ for a quotient polynomial $\hat{g}(X)$ of degree at most $\mathsf{deg}(\hat{f}) - |S|$. Therefore, if $\hat{f}(X)$ is low-degree and agrees with p then $\mathsf{Quotient}(\hat{f}, S, p)(X)$ is a low-degree polynomial. On the other hand, the powerful lemma below roughly says the opposite: if all low-degree polynomials close to a function f over domain L disagree with p, then the quotient function $\mathsf{Quotient}(f, S, p)$ is far from low-degree over L.

Lemma 2. *Suppose that for every polynomial $\hat{f}$ of degree at most d with $\Delta(f, \hat{f}(L)) \leq \delta$ it holds that there exists $a \in S$ where $\hat{f}(a) \neq p(a)$.*[6] *Then $\mathsf{Quotient}(f, S, p)$ is δ-far on L from every polynomial of degree $d - |S|$.*

Proof sketch. Let $g := \mathsf{Quotient}(f, S, p)$ and suppose towards contradiction that there exists a polynomial $\hat{g}$ of degree at most $d - |S|$ that agrees with g on at least a $(1 - \delta)$-fraction of the locations of L. Consider the "unquotiented" polynomial $\hat{f}(X) = \hat{V}_S(X) \cdot \hat{g}(X) + \hat{p}(X)$, which has degree at most d. For every x where $\hat{g}(x) = g(x)$, we have

$$\hat{f}(x) = \hat{V}_S(x) \cdot \hat{g}(x) + \hat{p}(x) = \hat{V}_S(x) \cdot g(x) + \hat{p}(x) = f(x) \ .$$

It follows that f is δ-close to $\hat{f}$ on L. Moreover, $\hat{f}(a) = \hat{p}(a) = p(a)$ for every $a \in S$. This is a contradiction to the assumption in the lemma statement. □

Let $\delta := 1 - \rho^{1/\Omega(1)}$. Suppose momentarily that for every f_i the verifier has a pair (x_i, y_i) such that there is *at most one* polynomial $\hat{f}_i$ that is δ-close to f_i and has $\hat{f}_i(x_i) = y_i$.

Under this assumption we use Lemma 2 to solve the issues with the naive compiler. Suppose that, following the interaction, the poly-IOP verifier queries $\hat{f}_i$ at locations $S_i \subseteq \mathbb{F}$. The prover sends to the verifier a function $p_i\colon S_i \to \mathbb{F}$, where, in the honest case $p_i(t) = \hat{f}_i(t)$ is the result of querying $\hat{f}_i$ at t. The verifier checks that these query answers cause the poly-IOP verifier to accept and then,

[6] For a function $f\colon D \to \mathbb{F}$ and set $L \subseteq D$, $f(L)$ is the restriction of f to L.

rather than testing proximity of f_i to low-degree, tests the proximity of $g_i := \mathsf{Quotient}(f_i, S'_i, p'_i)$ where $S'_i := S_i \cup \{x_i\}$ and $p'_i \colon S'_i \to \mathbb{F}$ has $p'_i(t) = p_i(t)$ for every $t \in S_i$ and $p'_i(x_i) = y_i$. Observe that the verifier can compute the value of g_i at any point in L because it knows S'_i and p'_i and has oracle access to f_i.

By Lemma 2, if for every polynomial that is δ-close to f_i there exists a point in S'_i where this polynomial disagrees with p'_i, then g_i is δ-far from low-degree. Since, by assumption, there is at most one polynomial $\hat{f}_i$ that is δ-close to f_i and $\hat{f}_i(x_i) = y_i$, the only way for the prover to keep g_i from being very far from low-degree is to give answers p_i that are consistent with this single polynomial (if there are no polynomials consistent with (x_i, y_i) then no matter what the prover does, g_i will be very far from low-degree). If this is the case for every f_i, it follows that for every f_i the query answers sent by the prover are consistent with exactly one polynomial, and so soundness holds by security of the original poly-IOP system.

It remains to discuss how to generate the pairs $((x_i, y_i))_{i \in [\mathsf{k}]}$.

Out-of-domain sampling. For every f_i we wish to generate (x_i, y_i) such that there is at most one polynomial $\hat{f}_i$ that is $(1 - \rho^{1/\Omega(1)})$-close to f_i and has $\hat{f}_i(x_i) = y_i$. For this we use an "out-of-domain sampling" technique, based on techniques of [BGKS20]. We emulate the interaction of the poly-IOPP where, as before, in round i when the prover is supposed to send a polynomial $\hat{f}_i$ it instead sends $f_i \colon L \to \mathbb{F}$, the evaluation of $\hat{f}_i$ over L. Immediately following the prover's message the verifier samples a random point $x_i \leftarrow \mathbb{F}$ and the prover replies with $y_i \in \mathbb{F}$ where (in the honest case) $y_i := \hat{f}_i(x_i)$. The verifier then samples its random message as in the poly-IOPP, and the protocol continues to the next round.

The Reed–Solomon code $\mathsf{RS}[\mathbb{F}, L, d_i]$, which f_i allegedly belongs to, is (δ, ℓ)-list decodable for $\delta := 1 - \rho^{1/\Omega(1)}$ and $\ell := \mathrm{poly}(1/\rho)$.[7] Since any pair of polynomials of degree at most d_i can agree on at most d_i points, with probability at least $1 - \binom{\ell}{2} \cdot \frac{d_i}{|\mathbb{F}|} = 1 - \frac{d_i \cdot \mathrm{poly}(1/\rho)}{|\mathbb{F}|}$ there is no pair of polynomials in the list-decoding of f_i whose evaluations on x_i are equal. If this is the case, then for every y_i sent by the prover y_i is consistent with at most one degree d_i polynomial that is δ-close to f_i on L. Thus, except with very small probability, we have generated the desired pair (x_i, y_i).

Put together with the technique of quotienting, in summary, either for every i all of the queries made by the verifier of the poly-IOP to the i-th polynomial are answered consistently with a single polynomial $\hat{f}_i$, in which case proximity holds by the properties of the poly-IOP, or at least one of the (quotiented) functions f_i is $(1 - \rho^{1/\Omega(1)})$-far from its supposed degree d_i, which is caught by the low-degree test.

Reducing to testing a single function. As described so far, the transformation needs to test for each f_i that its quotiented function is close to low-degree. Since the low-degree test is typically the least efficient part of the the proof, we would like to run it only once (in fact, saving this factor will be crucial to eliminate blow-up in the recursive way that we use this construction). To solve this problem we instead test the proximity of a random linear combination of the functions $g_1, \ldots, g_{\mathsf{k}}$. [BCIKS20] show that the random linear combinations of functions preserves the maximum distance of each one of the functions from their respective codes.

A technical issue that arises is that the functions being tested are of different degrees, a setting not considered in [BCIKS20]. This issue is addressed in [BCRSVW19] by appropriately shifting the degrees of the functions to match the maximum degree among them; however, their analysis is limited to constant soundness. In contrast, we provide a new high-soundness analysis, by leveraging the correlated agreement of the original functions and their shifted versions.

[7] A code is (δ, ℓ)-list decodable if for every function f there are at most ℓ codewords at distance at most δ from f.

2.1.1 Summary: compiling poly-IOPs to IOPs

We summarize our compiler from poly-IOP to IOP. Let $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$ be the prover and verifier of a k-round poly-IOP. In this summary, for simplicity, we assume that every polynomial $\hat{f}_i$ sent by the poly-IOP prover has the same degree d, and that the poly-IOP verifier makes exactly q queries to each of the polynomials sent by the prover (so that $|S_i| = \mathsf{q}$ for every i). The IOP is as follows.

1. For $i = 1, \ldots, \mathsf{k}$:
   (a) The prover runs $\mathbf{P}_{\mathsf{PIOP}}$ to obtain a polynomial $\hat{f}_i \in \mathbb{F}^{\leq d}[X]$, and sends $f_i \colon L \to \mathbb{F}$ (the evaluation of $\hat{f}_i$ on L).
   (b) The verifier samples $x_i \leftarrow \mathbb{F}$ uniformly at random.
   (c) The prover answers with $y_i := \hat{f}_i(x_i)$.
   (d) The verifier sends the message that $\mathbf{V}_{\mathsf{PIOP}}$ sends in the i-th round.
2. For every i, the prover computes the set of queries S_i that $\mathbf{V}_{\mathsf{PIOP}}$ needs to make to $\hat{f}_i$ and sends $p_i \colon S_i \to \mathbb{F}$ where $p_i(a) = \hat{f}_i(a)$ for every $a \in S_i$.
3. The verifier samples random coefficients $\xi_1, \ldots, \xi_{\mathsf{k}}$.
4. The prover and verifier run the proximity test for $\mathsf{RS}[\mathbb{F}, L, d - \mathsf{q} - 1]$ on the function
$$h(x) = \sum_{i \in [\mathsf{k}]} \xi_i \cdot g_i(x) \ ,$$
   where $g_i := \mathsf{Quotient}(f_i, S_i', p_i')$ for $S_i' := S_i \cup \{x_i\}$ and $p_i'(a) = p(a)$ if $a \in S_i$ and $p_i'(x_i) = y_i$. For a query t made by the proximity test verifier to h, the verifier computes $g_i(t)$ for every i using S_i', p_i' and its access to f_i and returns to the proximity test verifier the weighted sum of the results.
5. The verifier accepts if and only if $\mathbf{V}_{\mathsf{PIOP}}$ accepts given the query answers described by the functions $p_1, \ldots, p_{\mathsf{k}}$ and the proximity test verifier accepts.

We sketch the idea behind the proof of Lemma 1. As previously argued, with high probability no two polynomials that are of distance $\delta := 1 - \rho^{1/\Omega(1)}$ from f_i have the same output on the point x_i sent by the verifier. Therefore, with high probability there is at most one polynomial that is δ-close to f_i and whose evaluation on x_i is equal to y_i.

If the prover answers with y_i such that no polynomial that is δ-close to f_i evaluates to y_i on x_i, then by Lemma 2, g_i will be δ-far from low-degree and consequently so will h (with high probability). This will be caught with high probability during the proximity test. Thus we can safely assume that there is exactly one polynomial $\hat{f}_i$ that evaluates to y_i on x_i. At this point the prover sends query answers. It has two options:

- For some i send query answers that are inconsistent with $\hat{f}_i$: in this case, there is no polynomial that is δ-close to f_i that agrees with p_i' on every output. By Lemma 2 this implies that g_i is δ-far from low-degree, which will be caught by the proximity test.

- For every i send query answers that are consistent with $\hat{f}_i$: If the prover takes this strategy then soundness of the compiled IOP holds by the soundness of the poly-IOP. Indeed, suppose that the prover uses this strategy and causes the verifier to accept with high probability. Then we can use the prover's strategy to break security of the poly-IOP by list-decoding the message f_i sent by the prover and finding the (single) polynomial in the list that evaluates to y_i on x_i. This polynomial must be the same one that the compiled prover is consistent with. Since these polynomials cause the poly-IOP verifier to accept in the compiled protocol, they do so in the poly-IOP as well.

In both options the verifier will reject with high probability. We conclude that new IOP is sound.

2.1.2 Bonus: low-degree testing for all domains

IOPPs for RS codes typically work over specific domains with nice properties (e.g., smooth domains). Using our compiler we develop a "domain shifting" technique, that can be used to extend the applicability of *any* IOPPs for RS codes to work over *any* evaluation domain (provided that the field contains a nice domain). The transformation has negligible efficiency loss.

Consider the following straightforward poly-IOPP for testing proximity to the code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$: given a function f to be tested, the prover sends a polynomial $\hat{h} \in \mathbb{F}^{\leq d}[X]$. The verifier then picks $a \leftarrow L$ uniformly at random, and accepts if and only if $f(a) = \hat{h}(a)$.

One can readily see that the poly-IOPP has error $1-\delta$ when testing a δ-far function f. Moreover, this does not depend on the evaluation domain L. If we have an IOPP for testing RS codes over a different domain L', then we can apply Lemma 1 with the poly-IOPP for $\mathcal{C}$ using the IOPP for RS codes over L' to get an IOPP that works over L as desired. See Section 8.2 for more details.

2.2 poly-IOPP for RS codes

We sketch a poly-IOPP for the code $\mathsf{RS}[\mathbb{F}, L, d]$ (evaluations on the domain L of polynomials over the field $\mathbb{F}$ of degree at most d). The test works for every degree $d \in \mathbb{N}$ but requires a field that is "nice" (that has some useful structure): we assume that $\mathbb{F}$ contains a multiplicative subgroup G whose order is a power of two and where $\sqrt{|\mathbb{F}|} > |G| \geq \mathrm{poly}(d, |L|)$.

Lemma 3 (informal)**.** *There is a poly-IOPP for $\mathsf{RS}[\mathbb{F}, L, d]$ with perfect completeness, soundness error (roughly) $2\sqrt{1-\delta} + \mathrm{poly}(d, 1/\rho)/|\mathbb{F}|$ for δ-far functions, round complexity $O(1)$, and total query complexity $O(1)$; here $\rho := (d+1)/|L|$. The prover sends $O(|L|/\rho)$ polynomials whose degrees are at most $O(\sqrt{d})$.*

In this sketch, we assume for simplicity that $d := m^2 - 1$ for some $m \in \mathbb{N}$ and that L is a multiplicative subgroup of $\mathbb{F}^*$ whose order is divisible by m. Our protocol, however, works for any degree d and evaluation domain L.

Our starting point is inspired by the PCPP of Ben-Sasson and Sudan [BS08]. We consider a mapping of a univariate polynomial into a bivariate polynomial obtained via the following fact.

Fact 1 ([BS06])**.** *For every $\hat{f} \in \mathbb{F}^{\leq d}[X]$ where $d := m^2 - 1$ and $\hat{q} \in \mathbb{F}^{\leq m}[X]$ there exists a unique bivariate polynomial $\hat{Q} \in \mathbb{F}[X, Y]$ with $\mathsf{deg}_X(\hat{Q}) = m - 1$ and $\mathsf{deg}_Y(\hat{Q}) \leq m - 1$ such that $\hat{f}(Z) = \hat{Q}(\hat{q}(Z), Z)$.*

Fix $\hat{q}(Z) := Z^m$. Define $d^* := m - 1$ and let $D \subseteq \mathbb{F} \times \mathbb{F}$ be the set of agreement points between $\hat{f}$ and $\hat{Q}$: $D := \{(\hat{q}(j), j) \mid j \in L\}$.

Let $f \colon L \to \mathbb{F}$ be a function claimed to be in $\mathsf{RS}[\mathbb{F}, L, d]$ and $Q \colon D \to \mathbb{F}$ be the bivariate function such that, for every $x \in L$, $Q(\hat{q}(x), x) = f(x)$. If f is the evaluation of a polynomial of degree d then, by Fact 1, Q is the evaluation of a bivariate polynomial of individual degree d^* (in both variables). On the other hand, if Q agrees with a bivariate polynomial $\hat{Q}$ of individual degree d^* on a $(1-\delta)$-fraction of the points of D, then the polynomial $\hat{f}(X) := \hat{Q}(\hat{q}(X), X)$ of degree at most $d^* \cdot \mathsf{deg}(\hat{q}) + d^* = m^2 - 1 = d$ agrees with f on a $(1-\delta)$-fraction of the points of L (indeed, for a $(1-\delta)$-fraction of L we have $\hat{f}(x) = \hat{Q}(\hat{q}(x), x) = Q(\hat{q}(x), x) = f(x)$).

In other words, the problem of testing proximity to Reed–Solomon codes with degree d can be reformulated as the problem of testing proximity to bivariate Reed–Muller codes of individual degree $d^* = O(\sqrt{d})$, for a related function. Note, however, that the formulation that we get is a

relatively difficult one when compared to standard (bivariate) Reed–Muller testing. This is because the domain D for which the function is given has little structure to leverage. In particular, it is *not the typical Cartesian product of two sets*. We describe our approach for testing such domains next.

2.2.1 A proximity test for Q

We wish to test that a given function $Q\colon D \to \mathbb{F}$ is δ-close to a bivariate polynomial with individual degree at most d^*. Since we are in the poly-IOPP model, we can assume that messages sent by the prover are univariate polynomials of degree at most d^*. Let $\mu := \sqrt{1-\delta}$. By definition of D, for every $(i,j) \in D$ it holds that $f(j) = \hat{Q}(i,j)$. Define $L_X := \{i \mid \exists\, j \text{ s.t. } (i,j) \in D\}$ and, for every $i \in L_X$, let $L_Y^{(i)} := \{j \mid (i,j) \in D\}$ (i.e., $L_Y^{(i)}$ is the set of all elements $j \in L$ such that $j^m = i$ over the multiplicative subgroup L). Since L (as described in the previous section) is a multiplicative subgroup whose order is divisible by m, each set $L_Y^{(i)}$ has size $|D|/|L_X|$.

Our protocol begins with the prover sending the rows of the polynomial $\hat{Q}$ corresponding to the domains $L_Y^{(i)}$; each row is a polynomial of degree d^*. That is, the prover sends polynomials $(\hat{r}_i)_{i \in L_X}$ where $\hat{r}_i(X) := \hat{Q}(i,X)$. These rows define corresponding columns (viewing the rows as a matrix and now looking at its columns). Specifically, we define $(c_j)_{j \in \mathbb{F}}$ where $c_j\colon L_X \to \mathbb{F}$ is defined as $c_j(i) := \hat{r}_i(j)$ for $i \in L_X$. If the prover is honest, then c_j is the evaluation of $\hat{Q}(\cdot, j) \in \mathbb{F}^{\leq d^*}[X]$ over L_X, and so $c_j \in \mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, d^*]$. Let $M(i,j) = \hat{r}_i(j) = c_j(i)$ for $i \in L_X$ and $j \in \mathbb{F}$ be the description of the rows and columns as a matrix on $L_X \times \mathbb{F}$.

While the rows of M must be low-degree polynomials (we are in the poly-IOPP model), it may be that they do not agree with a low-degree bivariate polynomial or that they are inconsistent with the function Q whose proximity to low-degree we are trying to test (i.e., there are many rows $\hat{r}_i$ where $\hat{r}_i(j) \neq Q(i,j)$ for many $j \in L_Y^{(i)}$), so we have to test both properties. Moreover, we must test for both properties simultaneously. That is, if we know that the rows of M are μ-close to low-degree and that the rows of M are μ-close to Q, we *cannot* conclude Q is close to low-degree. Indeed, in our regime μ is small (much smaller than $1/2$), and thus the μ fraction for which M is close to low-degree might be disjoint from the μ fraction that M close to Q.

In Section 2.2.2 we explain how we test consistency of the rows with a low-degree bivariate polynomial and then in Section 2.2.3 we explain how to test consistency with Q in the same area.

2.2.2 Consistency with low-degree bivariate polynomial

We discuss how to test that the rows (and the matrix M constructed by them) are consistent with a low-degree bivariate polynomial. Recall that the rows of M are low-degree polynomials (since we are in the poly-IOPP model). We therefore need only show that the columns of M are consistent with low-degree polynomials.

A natural approach to test that the columns of M are close to low-degree is to choose a random column c_j and test that it is low-degree. Since we are in the poly-IOP model, and the column c_j is meant to be the evaluation of a degree d^* polynomial, this test can be done straightforwardly: the prover sends to the verifier a polynomial $\hat{v}_j \in \mathbb{F}^{\leq d^*}[X]$ that (in the honest case) is supposed to be the polynomial whose evaluation over L_X is equal to c_j (i.e., $\hat{v}_j(a) = c_j(a)$ for every $a \in L_X$). The verifier samples a random point $a \leftarrow L_X$ and checks that $\hat{v}_j(a) = c_j(a) = \hat{r}_a(j)$ by sampling $\hat{v}_j$ and $\hat{r}_a(j)$. Since $\hat{v}_j$ is a low-degree polynomial, passing this test ensures that c_j is close to low-degree.

We next assess whether the test of choosing a random column and testing that it is close to low-degree, as described above, suffices to show that M is close to a low-degree bivariate polynomial.

Does the naive approach suffice? Polishchuck and Spielman [PS94] show that if, with (large) constant probability, a random c_j is low-degree then there is a low-degree bivariate polynomial $\hat{Q}$ that is at most a constant-far from M, and so a constant fraction of the rows $(\hat{r}_i)_{i \in L_X}$ are close to the rows of a low-degree polynomial. In more detail, they show that there exists a constant $\epsilon \in (0,1)$ such that if M is δ-far from low degree (for small enough constant δ) then the verifier accepts in the previously described test with probability at most $1 - (\epsilon \cdot \delta)$.

However their techniques do not apply when the verifier's acceptance probability nears $1/2$, so they are insufficient (as we want to accept with probability at most $\rho^{1/\Omega(1)}$ if M is $1 - \rho^{1/\Omega(1)}$ far from low-degree). Moreover, even if one would extend these results for any regime, the proximity that they achieve does not suffice for our needs. Indeed, if δ approaches 1 (as in the high-soundness regime) then the [PS94] result would only be able to conclude that the verifier accepts with at most a constant probability $1 - \epsilon$.

Chiesa, Manohar, and Shinkar [CMS20] extend the analysis of this test to the low-error regime, but at the cost of an exponentially-large field and weakening the result to a "list-decoding"-style claim. This is also insufficient for us, as we require that the field size be polynomial. It is currently unknown whether this test holds in the low-agreement, small-field regime with a suitable error term.

A standard approach to reduce the error is to repeat the test multiple times (i.e., test multiple columns). Doing this naively using repetition to achieve the error bound that we want would highly increase the query complexity. Nonetheless, we manage to test that many columns are low-degree in *one shot* with constant query complexity, as we explain below.

Proximity gaps. Consider a "nice" domain $H \subseteq \mathbb{F}$ of order d^*. We test all of the columns specified by H at once with $O(1)$ queries (in the poly-IOPP model). We use the recent analysis of *proximity gaps* of Reed–Solomon codes [BCIKS20]. If, with sufficient probability, a random linear combination of the columns $(c_j)_{j \in H}$ is close to the code $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, d^*]$, then the columns $(c_j)_{j \in H}$ are all close to $\mathcal{C}_X$. In fact, they have the stronger property of *correlated agreement*: there is a set $W \subseteq L_X$ such that for every $j \in H$ there exists $u_j \in \mathcal{C}_X$ where $c_j(W) = u_j(W)$. For coefficients $\vec{\xi} = (\xi_j)_{j \in H}$ let $c_{\vec{\xi}} = \sum_{j \in H} \xi_j \cdot c_j$, and let $\Delta(f, g)$ denote the relative Hamming distance between f and g.

Theorem 4 (Proximity gap for RS codes [BCIKS20]; informal)**.** *If* $\Pr_{\vec{\xi}}[\Delta(c_{\vec{\xi}}, \mathcal{C}_X) \leq 1 - \mu] > \varepsilon_{\mathsf{gen}} = O(\mathrm{poly}(|L_X|)/|\mathbb{F}|)$ *then there exists a set* $W \subseteq L_X$ *of size at least* $\mu \cdot |L_X|$ *such that for every* $j \in H$ *there exists* $u_j \in \mathcal{C}_X$ *where* $c_j(W) = u_j(W)$.

Thus, if we test that $c_{\vec{\xi}}$ is low-degree (by the same technique as we did for c_j) for a random vector of coefficients, then we can ensure that all of the columns are low-degree (in a correlated manner). By the following claim, this will suffice for us to prove proximity of M to the evaluation of a low-degree bivariate polynomial in the low-error regime.

Claim 1. *If there exists* $W \subseteq L_X$ *on which* $(c_j)_{j \in H}$ *have correlated agreement and* $|W| \geq \mu \cdot |L_X| \geq d^*$*, then there exists a bivariate polynomial* $\hat{Q} \in \mathbb{F}[X, Y]$ *with individual degree bounded by* d^* *such that* $\hat{Q}(i, Z) = M(i, Z) = \hat{r}_i(Z)$ *for every* $i \in W$.

We have reduced the problem of checking proximity to the rows of a low-degree bivariate polynomial to the following test: The verifier chooses random coefficients $\vec{\xi}$, and the prover sends a polynomial $\hat{v}$ claimed to be equal to the low-degree extension of the column $c_{\vec{\xi}}$. The verifier compares the two at a random location in L_X. This requires the verifier to compute $c_{\vec{\xi}}(i)$ at some point i. Naively, the verifier could query each column and check that their weighted sum in this location is equal to $\hat{v}(i)$. This would have large query complexity, so we must find another way.

Univariate sumcheck. We observe that the univariate sumcheck technique and the proximity gaps go hand-in-hand: notice that $c_{\vec{\xi}}(i) = \sum_{j \in H} \xi_j \cdot c_j(i) = \sum_{j \in H} \xi_j \cdot \hat{r}_i(j)$. Therefore, in order to compute the weighted sum of the columns at an index i, it suffices to compute the weighted sum of evaluations of $\hat{r}_i$ on H. To compute this sum, we use a (weighted) univariate sumcheck protocol, first described in [BCRSVW19].

In a univariate sumcheck protocol, the prover proves that $\sum_{j \in H} \xi_j \cdot \hat{r}_i(j) = \gamma$, provided that the verifier has access to $\hat{r}_i$, and provided that H is a multiplicative subgroup of $\mathbb{F}^*$,[8] both of which can be guaranteed. Using the univariate sumcheck protocol, we can complete our test. The standard univariate sumcheck protocol *assumes the existence of a low-degree test*, which is a circular dependency in our case. However, it requires this low-degree test to hold for (roughly) the degree of the polynomial being summed over. Thus, if we only perform the sumcheck protocol for degree d^* polynomials, then we can use the poly-IOP model for this (we remind the reader at this point that the poly-IOP model will be compiled recursively to a standard IOPP).

2.2.3 Ensuring simultaneous consistency with Q

So far, we have forced the prover to send rows $(\hat{r}_i)_{i \in L_X}$ that are consistent with some low-degree bivariate polynomial $\hat{Q}$. The prover could still send rows that are inconsistent with Q (and are consequently inconsistent with f). Testing this property is relatively straightforward: the verifier chooses a row $\hat{r}_i$ at random and compares it with Q on a random point in $L_Y^{(i)}$ (the locations on which $\hat{r}_i$ and Q coincide).

If many rows $(\hat{r}_i)_{i \in L_X}$ have consistency with Q on only a small fraction of the locations, then the verifier will reject with high probability in this test. Moreover, since every set $L_Y^{(i)}$ has the same number of points, sampling a uniform $i \in L_X$ and then uniform $j \in L_Y^{(i)}$ is identical to sampling a uniformly random $(i, j) \in D$ (recall that $D \subseteq \mathbb{F} \times \mathbb{F}$ is the evaluation domain over which Q is defined). Thus if this test passes with probability μ, it means that $\Delta(M(D), Q) \leq 1 - \mu$ where $M(D)$ is the matrix M described by the rows $\hat{r}_i$ restricted to the locations in D.

Unfortunately, even though (by the previous section) a μ-fraction of the rows of M are identical to the rows of a low-degree bivariate polynomial $\hat{Q}$, we cannot infer that Q is close to $\hat{Q}$. Relying on Theorem 4, we can only infer that a (possibly small) fraction μ of the rows are consistent with $\hat{Q}$. It may be the case that all of the rows that are consistent with Q are inconsistent with $\hat{Q}$. This is exemplified by the fact that, by applying the triangle inequality, we can only conclude that Q has distance at most $2 \cdot (1 - \mu)$ from $\hat{Q}$, which is only non-trivial if $\mu > 1/2$, whereas we need to handle the case of very small μ.

To solve this, the verifier could test the same row i for consistency with Q and with $\hat{Q}$ (i.e., test consistency with Q on the same row i that we run the univariate sumcheck from before). Even this is not enough. The reason is that Theorem 4 only says that W exists, and says nothing about which indices are contained within it. In particular, it may be the case that the rows in W are inconsistent with Q i.e., it may be that $\hat{r}_i(j) \neq Q(i, j)$ for every $i \in W$ and $j \in L_Y^{(i)}$. Thus we cannot conclude that Q is close to the bivariate polynomial $\hat{Q}$. To bypass this issue, we prove a stronger version of Theorem 4, which will give us this extra guarantee, which we call proximity gaps for subsets.

Proximity gaps for subsets. Our strengthening of Theorem 4 essentially shows that if (with probability $\varepsilon_{\mathsf{gen}}$) a random linear combination agrees with a codeword on a fixed set S, then there

[8] Univariate sumcheck protocols are also known for other summation domains such as additive subgroups (see, e.g., [BCRSVW19]), but in this paper, we only use the multiplicative subgroup version.

exists W as in Theorem 4, with the additional guarantee that $W \subseteq S$.

Theorem 5 (Theorem 4.4; informal)**.** *Fix $S \subseteq L_X$ with $|S| \geq \mu \cdot |L_X|$. If*

$$\Pr_{\xi_1,\ldots,\xi_{|H|}}\left[\begin{array}{ll}\exists\, T \subseteq S & \\ |T| \geq \mu \cdot |L_X| & : \quad \exists\, u \in \mathcal{C}_X,\ u(T) = \sum_{j\in H} \xi_j \cdot c_j(T)\end{array}\right] > \varepsilon_{\mathsf{gen}}$$

then there exists a set $W \subseteq S$ of size at least $\mu \cdot |L_X|$ such that for every $j \in H$ there exists $u_j \in \mathcal{C}_X$ where $c_j(W) = u_j(W)$.

Plugging in as S the set of all rows for which there is μ-agreement with Q, we now have what we wanted: there is a large set W where the columns $(c_j)_{j\in H}$ have correlated agreement and also μ-agreement with Q. Then, each one of the $\mu \cdot |L_X|$ rows $\hat{r}_i$ in W has at least $\mu \cdot |L_Y^{(i)}| = \mu \cdot |D|/|L_X|$ locations where $\hat{r}_i$ agrees with Q. By Claim 1, this applies to $\hat{Q}$ as well. It follows that Q agrees with $\hat{Q}$ on at least $\mu^2 = 1-\delta$ of the points in D, and so Q (and consequently our original univariate function f) is δ-close to low-degree. This suffices for our proof.

2.2.4 Summary: poly-IOPP for RS codes

Below we summarize the poly-IOPP underlying our Lemma 3. Recall that we begin with a univariate function f and deduce from it a bivariate function $Q\colon D \to \mathbb{F}$ (where $D := \cup_{i\in L_X}(\{i\} \times L_Y^{(i)})$) on which we test proximity.

1. The prover sends $(\hat{r}_i)_{i\in L_X}$ where $\hat{r}_i(X) := \hat{Q}(i, X) \in \mathbb{F}^{\leq d^*}[X]$.
2. The verifier samples random coefficients $(\xi_j)_{j\in H}$.
3. The prover sends $\hat{v} \in \mathbb{F}^{\leq d^*}[X]$, where in the honest case $\hat{v}(i) = \sum_{j\in H} \xi_j \cdot \hat{r}_i(j)$ for every $i \in L_X$.
4. The verifier samples a random $i \leftarrow L_X$.
5. The prover and verifier run the weighted univariate sumcheck protocol for "$\hat{v}(i) = \sum_{j\in H} \xi_j \cdot \hat{r}_i(j)$".
6. The verifier samples a random $j \leftarrow L_Y^{(i)}$, and accepts if and only if the sumcheck verifier accepted and $\hat{r}_i(j) = Q(i, j)$.

Proof sketch of Lemma 3. Let $\mu = \sqrt{1-\delta}$. Suppose that the verifier accepts in the above poly-IOPP with probability greater than $2\mu + \varepsilon_{\mathsf{gen}}$. Let the set S be the indices of the rows for which $\hat{r}_i$ and Q agree on (at least) a μ-fraction of $L_Y^{(i)}$ (the points on which they coincide), and let $T \subseteq S$ be the rows in S that also have $\hat{v}(i) = \sum_{j\in H} \xi_j \cdot \hat{r}_i(j)$.

We bound the probability that the verifier rejects when T is small: we argue that if $|T| < \mu \cdot |L_X|$ then the verifier accepts with probability at most 2μ. The verifier samples $i \leftarrow L_X$. With probability at most μ, it holds that $i \in T$. If $i \notin T$, either $\hat{v}(i) \neq \sum_{j\in H} \xi_j \cdot \hat{r}_i(j)$, in which case the verifier will reject in the univariate sumcheck, or $\hat{r}_i$ and Q agree on less than a μ-fraction of $L_Y^{(i)}$, so the verifier will choose a point on which they agree with probability at most μ.

Since, by assumption, the verifier accepts with probability at least $2\mu + \varepsilon_{\mathsf{gen}}$, it follows that $|T| \geq \mu \cdot |L_X|$ with probability greater than $\varepsilon_{\mathsf{gen}}$. Consequently, by Theorem 5, there exists $W \subseteq S$ with $|W| \geq \mu \cdot |L_X|$ where the columns $(c_j)_{j\in H}$ defined by the rows $(\hat{r}_i)_{i\in L_X}$ have correlated agreement. From Claim 1, we deduce that there exists a bivariate polynomial $\hat{Q}$ that agrees with a μ-fraction of rows, each of which agrees with Q on an μ-fraction of the locations. Therefore Q agrees with $\hat{Q}$ on at least $\mu^2 = 1-\delta$ of its points. □

We have shown that in the poly-IOPP model, we can test δ-proximity with proximity error (roughly) $2\sqrt{1-\delta} + \varepsilon_{\mathsf{gen}}$. See Section 6.2 for a formal statement and proof. In order to achieve our IOPP for univariate polynomials (Theorem 2), we still need to compile the poly-IOPP into a standard IOPP and apply recursion both of which add to the proximity error of the final protocol.

2.3 Testing RS codes with inverse polynomial error

We combine the tools developed in Section 2.1 and Section 2.2 in order to prove Theorem 2.

Let $f\colon L \to \mathbb{F}$ be a function claimed to be in $\mathsf{RS}[\mathbb{F}, L, d]$. We recursively reduce the degree by a square root until it is a constant. Proximity to $\mathsf{RS}[\mathbb{F}, L, O(1)]$ can then be checked directly.

For simplicity we assume that $d = 2^{2^t} - 1$ for $t \in \mathbb{N}$ (though this is not required in our construction) and that L is a multiplicative subgroup of $\mathbb{F}^*$. In step i of the recursion, let $d_i = 2^{2^{t-i}} - 1$ and L_i be an evaluation domain chosen such that $\mathsf{RS}[\mathbb{F}, L_i, d_i]$ has rate $\rho_i = (d_i+1)/|L_i| = (d+1)/|L| = \rho$. Assume that there exists a low-degree test for $\mathsf{RS}[\mathbb{F}, L, d_{i+1}]$. According to Lemma 3 there exists a poly-IOPP for $\mathsf{RS}[\mathbb{F}, L, d_i]$ where the prover's messages are degree d_{i+1} polynomials. We then combine the poly-IOPP with the proximity test for $\mathsf{RS}[\mathbb{F}, L, d_{i+1}]$ using the transformation of Lemma 1, whose output is a standard IOPP for $\mathsf{RS}[\mathbb{F}, L, d_i]$. The soundness error of the resulting IOPP is (roughly) $2\sqrt{1-\delta} + \beta_{\mathsf{prx}}(1-\rho^{1/\Omega(1)}) + \frac{\mathrm{poly}(|L|)}{|\mathbb{F}|}$, where $\beta_{\mathsf{prx}}(1-\rho^{1/\Omega(1)})$ is the soundness error of the proximity test for $\mathsf{RS}[\mathbb{F}, L, d_{i+1}]$ for $(1-\rho^{1/\Omega(1)})$-far functions.

There are $t = \mathrm{loglog}\, d$ levels of recursion. In each layer, the soundness error is increased by (very roughly) a $\rho^{1/\Omega(1)} + \mathrm{poly}(|L|)/|\mathbb{F}|$ factor. Hence, the soundness error of the resulting IOPP is $2\sqrt{1-\delta} + \mathrm{loglog}\, d \cdot (\rho^{1/\Omega(1)} + \mathrm{poly}(|L|)/|\mathbb{F}|)$. In a more general and tight analysis we can get the soundness error down to $\max\left\{2\sqrt{1-\delta},\ \rho^{1/\Omega(1)}\right\} + \frac{\mathrm{poly}(d,1/\rho)}{|\mathbb{F}|}$. In order to get soundness error $\max\{1-\delta, \rho^{1/\Omega(1)} + \frac{\mathrm{poly}(d,1/\rho)}{|\mathbb{F}|}\}$ and in order to get rid of the assumption that L is a smooth subgroup as in Theorem 2, we additionally utilize the domain-shifting technique described in Section 2.1.2. See Section 6 for a formal proof and tight analysis.

This concludes the overview of the proof of Theorem 2.

3 Preliminaries

Throughout the paper, we use the "hat" symbol over function when we want to emphasize that they are polynomials (e.g., $\hat{p}$). For two functions $f, g \colon L \to \mathbb{F}$ we use $\Delta(f, g)$ to denote the fractional Hamming distance between f and g (the fraction of points in which they disagree).

3.1 Interactive oracle proofs

Interactive Oracle Proofs (IOPs) [BCS16; RRR16] are information-theoretic proof systems that combine aspects of Interactive Proofs [Bab85; GMR89] and Probabilistically Checkable Proofs [BFLS91; FGLSS96; AS98; ALMSS98], and also generalize the notion of Interactive PCPs [KR08]. Below we describe *public-coin* IOPs.

Recall that a k-round public-coin IOP works as follows. In every round $i \in [\mathsf{k}]$, the verifier sends a uniformly random message ρ_i to the prover; then the prover sends a proof string π_i to the verifier. After k rounds of interaction, the verifier makes some queries to the proof strings $\pi_1, \ldots, \pi_{\mathsf{k}}$ sent by the prover, and then decides if to accept or to reject.

In more detail, let $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ be a tuple where $\mathbf{P}$ is an interactive algorithm, and $\mathbf{V}$ is an interactive oracle algorithm. We say that IOP is a *public-coin IOP* for a relation R with k rounds, perfect completeness, and soundness error β if the following holds.

- **Completeness.** For every $(\mathbb{x}, \mathbb{w}) \in R$,

$$\Pr_{\rho_1, \ldots, \rho_{\mathsf{k}}} \left[\mathbf{V}^{\pi_1, \ldots, \pi_{\mathsf{k}}}(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{r} \pi_1 \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}) \\ \vdots \\ \pi_{\mathsf{k}} \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}, \rho_1, \ldots, \rho_{\mathsf{k}}) \end{array} \right] = 1 \ .$$

  An alternative definition exists, which allows for completeness error.

- **Soundness.** For every $\mathbb{x} \notin L(R)$ and unbounded malicious prover $\tilde{\mathbf{P}}$,

$$\Pr_{\rho_1, \ldots, \rho_{\mathsf{k}}} \left[\mathbf{V}^{\pi_1, \ldots, \pi_{\mathsf{k}}}(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{r} \pi_1 \leftarrow \tilde{\mathbf{P}}(\rho_1) \\ \vdots \\ \pi_{\mathsf{k}} \leftarrow \tilde{\mathbf{P}}(\rho_1, \ldots, \rho_{\mathsf{k}}) \end{array} \right] \leq \beta \ .$$

For interactive (oracle) algorithms $\mathbf{A}$ and $\mathbf{B}$, we denote by $\langle \mathbf{A}(a), \mathbf{B}(b) \rangle(c)$ the random variable describing the output of $\mathbf{B}$ following the interaction between $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}$ is given private input a, $\mathbf{B}$ is given private input b and both parties are given joint input c.

Efficiency measures. We study several efficiency measures. All of these complexity measures are implicitly functions of the instance $\mathbb{x}$.

- *Rounds* k: The IOP has k rounds of interaction.
- *Alphabet* Σ *and alphabet size* λ: the symbols of each π_i come from the alphabet Σ, of size λ. In this paper, the alphabet will always be a field $\mathbb{F}$.
- *Proof length (per round)* l: the number of symbols in each proof π_i.
- *Queries* q: the number of bits read by the verifier from $\pi_1, \ldots, \pi_{\mathsf{k}}$.
- *Randomness* r: the verifier's i-th message ρ_i has length r_i and $\mathsf{r} := \sum_{i=1}^{\mathsf{k}} \mathsf{r}_i$ is the total number of random bits sent by the verifier.
- *Verifier time* vt: $\mathbf{V}$ runs in time vt measured in algebraic field operations.

- *Decision complexity* dt: Following the choice of queries, $\mathbf{V}$ runs in time dt to decide whether to accept or reject.

3.2 IOPs of proximity

Let $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ be a tuple where $\mathbf{P}$ is an interactive algorithm, and $\mathbf{V}$ is an interactive oracle algorithm. We say that IOP is a public-coin IOP *of proximity* for a relation $R = \{(\mathbb{x}, \mathbb{w})\}$ with k rounds, perfect completeness, and proximity error β if the following holds.

- **Completeness.** For every $(\mathbb{x}, \mathbb{w}) \in R$,

$$\Pr_{\rho_1,\ldots,\rho_{\mathsf{k}}}\left[\begin{array}{l|r} \mathbf{V}^{\mathbb{w},\pi_1,\ldots,\pi_{\mathsf{k}}}(\mathbb{x},\rho_1,\ldots,\rho_{\mathsf{k}}) = 1 & \pi_1 \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w}) \\ & \vdots \\ & \pi_{\mathsf{k}} \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w},\rho_1,\ldots,\rho_{\mathsf{k}}) \end{array}\right] = 1 \ .$$

  An alternative definition exists, which allows for completeness error.

- **Proximity.** For every $(\mathbb{x}, \mathbb{w})$ pair where $\mathbb{w}$ is δ-far in relative Hamming distance from any $\mathbb{w}'$ where $(\mathbb{x}, \mathbb{w}) \in R$ and any unbounded malicious prover $\tilde{\mathbf{P}}$,

$$\Pr_{\rho_1,\ldots,\rho_{\mathsf{k}}}\left[\begin{array}{l|r} \mathbf{V}^{\mathbb{w},\pi_1,\ldots,\pi_{\mathsf{k}}}(\mathbb{x},\rho_1,\ldots,\rho_{\mathsf{k}}) = 1 & \pi_1 \leftarrow \tilde{\mathbf{P}} \\ & \vdots \\ & \pi_{\mathsf{k}} \leftarrow \tilde{\mathbf{P}}(\rho_1,\ldots,\rho_{\mathsf{k}}) \end{array}\right] \leq \beta(\delta) \ .$$

An IOPP has the same parameters as an IOP, except that we let:
- q_f be the number of queries made to $\mathbb{w}$.
- q_π be the number of queries made to the proofs $\pi_1, \ldots, \pi_{\mathsf{k}}$.

Proximity testing. A proximity tester for a code $\mathcal{C}$ is an IOPP for the relation containing pairs $(\mathbb{x}, \mathbb{w})$ where $\mathbb{x}$ are code parameters (such as, for Reed–Solomon codes, the field size, the evaluation domain, etc.) and $\mathbb{w} \in \mathcal{C}$ is a codeword.

3.3 Polynomial IOPs and IOPPs

A polynomial IOP (poly-IOP) is an IOP $(\mathbf{P}, \mathbf{V})$ system where the prover (both honest and malicious) sends as its messages the evaluation of univariate polynomials over a field $\mathbb{F}$. In more detail, for every round i there is a prescribed list of m_i degrees $(d_{i,j})_{j\in[\mathsf{m}_i]}$ where $d_{i,j} \in \mathbb{N}$. During round i, the prover (both honest and malicious) outputs m_i polynomials by specifying their coefficients, where the j-th polynomial, $\hat{f}_{i,j} \in \mathbb{F}^{\leq d_{i,j}}[X]$ has degree at most $d_{i,j}$. The verifier is then given as a message $(\hat{f}_{i,j}(\mathbb{F}))_{j\in\mathsf{m}_i}$ where $\hat{f}_{i,j}(\mathbb{F})$ is the evaluation of $\hat{f}_{i,j}$ over the entire field $\mathbb{F}$. In more detail, $(\mathbf{P}, \mathbf{V})$ is a public-coin poly-IOP for a relation $R = \{(\mathbb{x}, \mathbb{w})\}$ with k rounds, perfect completeness, and proximity error β, where the prover sends m_i polynomials in round i with degrees $(d_{i,j})_{j\in[\mathsf{m}_i]}$ if the following holds.

- **Completeness.** For every $(\mathbb{x}, \mathbb{w}) \in R$,

$$\Pr_{\rho_1,\ldots,\rho_{\mathsf{k}}}\left[\mathbf{V}^{\pi_1,\ldots,\pi_{\mathsf{k}}}(\mathbb{x},\rho_1,\ldots,\rho_{\mathsf{k}})=1 \;\middle|\; \begin{array}{r} (c^{(\ell)}_{1,j})_{j\in[\mathsf{m}_1],\ell\in[d_{1,j}]} \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w}) \\ \forall j\in[\mathsf{m}_1].\ \hat{f}_{1,j}(X) := \sum_{\ell\in[d_{1,j}]} c^{(\ell)}_{1,j}\cdot X^{\ell-1} \\ \pi_1 := (\hat{f}_{1,j}(\mathbb{F}))_{j\in[\mathsf{m}_1]} \\ \vdots \\ (c^{(\ell)}_{\mathsf{k},j})_{j\in[\mathsf{m}_{\mathsf{k}}],\ell\in[d_{\mathsf{k},j}]} \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w},\rho_1,\ldots,\rho_{\mathsf{k}}) \\ \forall j\in[\mathsf{m}_{\mathsf{k}}].\ \hat{f}_{\mathsf{k},j}(X) := \sum_{\ell\in[d_{\mathsf{k},j}]} c^{(\ell)}_{\mathsf{k},j}\cdot X^{\ell-1} \\ \pi_1 := (\hat{f}_{\mathsf{k},j}(\mathbb{F}))_{j\in[\mathsf{m}_{\mathsf{k}}]} \end{array}\right] = 1 \ .$$

  An alternative definition exists, which allows for completeness error.

- **Soundness.** For every $\mathbb{x} \notin L(R)$ and unbounded malicious prover $\tilde{\mathbf{P}}$,

$$\Pr_{\rho_1,\ldots,\rho_{\mathsf{k}}}\left[\mathbf{V}^{\pi_1,\ldots,\pi_{\mathsf{k}}}(\mathbb{x},\rho_1,\ldots,\rho_{\mathsf{k}})=1 \;\middle|\; \begin{array}{r} (c^{(\ell)}_{1,j})_{j\in[\mathsf{m}_1],\ell\in[d_{1,j}]} \leftarrow \tilde{\mathbf{P}} \\ \forall j\in[\mathsf{m}_1].\ \hat{f}_{1,j}(X) := \sum_{\ell\in[d_{1,j}]} c^{(\ell)}_{1,j}\cdot X^{\ell-1} \\ \pi_1 := (\hat{f}_{1,j}(\mathbb{F}))_{j\in[\mathsf{m}_1]} \\ \vdots \\ (c^{(\ell)}_{\mathsf{k},j})_{j\in[\mathsf{m}_{\mathsf{k}}],\ell\in[d_{\mathsf{k},j}]} \leftarrow \tilde{\mathbf{P}}(\rho_1,\ldots,\rho_{\mathsf{k}}) \\ \forall j\in[\mathsf{m}_{\mathsf{k}}].\ \hat{f}_{\mathsf{k},j}(X) := \sum_{\ell\in[d_{\mathsf{k},j}]} c^{(\ell)}_{\mathsf{k},j}\cdot X^{\ell-1} \\ \pi_1 := (\hat{f}_{\mathsf{k},j}(\mathbb{F}))_{j\in[\mathsf{m}_{\mathsf{k}}]} \end{array}\right] \leq \beta \ .$$

A poly-IOP has the same parameters as an IOP, except that, rather than counting the proof length, we count the number of functions:

- m is the number of polynomials sent by the prover: $\mathsf{m} := \sum_{i=1}^{\mathsf{k}} \mathsf{m}_i$.
- $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}}$ is the number of polynomials queried by the verifier (multiple queries to the same polynomial do not add towards this value). Observe that $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} \leq \mathsf{q}$.

When referring to the prover's messages we will generally ignore the description of the polynomials $\hat{f}_{i,j}$ as coefficients, and simply say that the prover outputs a polynomial. Similarly, since the verifier has oracle access to $\hat{f}_{i,j}$ evaluated over the entire field, we will simply denote that it has direct oracle access to $\hat{f}_{i,j}$.

We will also use the polynomial IOPs of proximity (poly-IOPP), which is defined similarly with respect to IOPPs. Note that the input to the poly-IOPP is not part of the prover's messages, and so is not required to be the evaluation of a polynomial over the field $\mathbb{F}$.

3.4 The Reed–Solomon and Reed–Muller codes

We first define error correcting codes.

Definition 3.1. *An* **error-correcting code** *of length n over an alphabet Σ is $\mathcal{C} \subseteq \Sigma^n$. We say that $\mathcal{C}$ is a* linear code *if $\Sigma = \mathbb{F}$ is a field and $\mathcal{C}$ is a subspace of $\mathbb{F}^n$.*

We now define the Reed–Solomon code:

Definition 3.2. *The* **Reed–Solomon code** *over field $\mathbb{F}$, evaluation domain $L \subseteq \mathbb{F}$ and degree $d \in \mathbb{N}$ is the set of evaluations over L of univariate polynomials (over $\mathbb{F}$) of degree at most d:*

$$\mathsf{RS}[\mathbb{F}, L, d] := \left\{ f \colon L \to \mathbb{F} \;\middle|\; \exists\, \hat{f} \in \mathbb{F}^{\leq d}[X] \text{ s.t. } \forall\, x \in L\,,\ f(x) = \hat{f}(x) \right\} \ .$$

The rate of $\mathcal{C}$ is $\rho := (d+1)/|L|$.

Given a code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ and function $f \colon L \to \mathbb{F}$, we sometimes use $\hat{f} \in \mathbb{F}^{\leq d}[X]$ to denote the nearest polynomial to f on L.

Definition 3.3. *For a code $\mathcal{C} \subseteq \mathbb{F}^n$, parameter $\gamma \in [0,1]$, and $v \in \mathbb{F}^n$, $\mathsf{List}_{\mathcal{C},\gamma}(v)$ denotes the list of codewords of $\mathcal{C}$ within relative Hamming distance at most γ from v.*

*We say that $\mathcal{C}$ is (γ, ℓ)–***list decodable** *if $|\mathsf{List}_{\mathcal{C},\gamma}(v)| \leq \ell$ for every $v \in \mathbb{F}^n$.*

The following fact says that if a code is list-decodable, then any code that is a subset of it is list-decodable with the same parameters.

Fact 3.4. *If a code $\mathcal{C} \subseteq \mathbb{F}^n$ is (γ, ℓ)–list decodable, then any $\mathcal{C}' \subseteq \mathcal{C}$ is (γ, ℓ)–list decodable.*

The Johnson bound shows that the Reed–Solomon code is list-decodable:

Theorem 3.5 (Johnson bound)**.** *For every $\eta \in (0, 1 - \sqrt{\rho})$, the Reed–Solomon code $\mathsf{RS}[\mathbb{F}, L, d]$ is $(1 - \sqrt{\rho} - \eta, 1/(2\eta\sqrt{\rho}))$–list decodable, where ρ is the rate of the code.*

As part of our construction we will need standard IOPPs for testing the Reed–Solomon code when the degree is constant:

Claim 3.6. *Let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ be a Reed–Solomon code where $d = O(1)$ is a constant. There exists a proximity test for $\mathcal{C}$ with the following parameters:*

Proximity test to show δ proximity to $\mathcal{C}$ with constant degree	
Proximity error	$1 - \delta$
Rounds	1
Alphabet	$\mathbb{F}$
Proof length	$O(1)$
Input queries	1
Proof queries	$O(1)$
Randomness	$\log \lvert\mathbb{F}\rvert$
Verifier running time	$O(1)$

Finally, we define the bivariate Reed–Muller code:

Definition 3.7. *The (individual-degree bivariate)* **Reed–Muller code** *over field $\mathbb{F}$, evaluation domain $D \subseteq \mathbb{F} \times \mathbb{F}$ and degrees $d_X, d_Y \in \mathbb{N}$ is the set of evaluations over D of bivariate polynomials (over $\mathbb{F} \times \mathbb{F}$) of degree at most d_X in their first variable, and degree at most d_Y in their second variable:*

$$\mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)] := \left\{ f \colon D \to \mathbb{F} \;\middle|\; \exists\, \hat{f} \in \mathbb{F}^{\leq d_X, \leq d_Y}[X, Y] \text{ s.t. } \forall\, (x,y) \in D\,,\ f(x,y) = \hat{f}(x,y) \right\} \ .$$

3.5 Polynomial identity lemma

We extensively use the polynomial identity lemma, whose various forms are credited to (at the very least) Schwartz, Zippel, and DeMillo and Lipton.

Lemma 3.8. *For any non-zero polynomial $\hat{p} \in \mathbb{F}^{\leq d}[X]$ it holds that $\Pr_{a \leftarrow \mathbb{F}}[\, \hat{p}(a) = 0 \,] \leq d/|\mathbb{F}|$.*

4 Proximity generators for correlated agreement

In this section we define proximity generators, and prove facts about them.

- In Section 4.1 we define proximity generators and rehash a theorem of Ben-Sasson et al. [BCIKS20] that shows that if it is likely that the random linear combination of functions is close to a Reed–Solomon codeword, then these vectors have correlated agreement with the Reed–Solomon code.
- In Section 4.2, we show that this is true also when fixed on a specific subset inside the evaluation domain. We call generators with this property *strong* proximity generators.

4.1 Proximity generators

A proximity generator generates coefficients whose linear combination with functions $f_1, \ldots, f_{\mathsf{m}}$ preserves their correlated agreement with a code.

Definition 4.1. *Let $\mathcal{C} \subseteq \mathbb{F}^\ell$ be a code. We say that $\mathcal{C}$ has a **proximity generator** for m functions with seed length w, proximity bound ψ, error ε, and computation time t, if there exists an t-time computable function $\mathsf{Gen}\colon \{0,1\}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{m}}$ such that for every $\delta \in (0, 1-\psi)$ and functions $f_1 \ldots, f_{\mathsf{m}}\colon [\ell] \to \mathbb{F}$, if*

$$\Pr\left[\Delta\left(\sum_{i\in[\mathsf{m}]} \xi_i \cdot f_i, \mathcal{C}\right) \leq \delta \;\middle|\; \begin{array}{r} s \leftarrow \{0,1\}^{\mathsf{w}} \\ (\xi_1, \ldots \xi_{\mathsf{m}}) \leftarrow \mathsf{Gen}(s) \end{array}\right] > \varepsilon(\delta) \ ,$$

then there exists $S \subseteq [\ell]$ with $|S| \geq (1-\delta)\cdot\ell$, and

$$\forall i \in [\mathsf{m}],\ \exists u \in \mathcal{C},\ f_i(S) = u(S) \ .$$

Throughout this paper we use the following claim, showing the Reed–Solomon codes have good proximity generators.

Theorem 4.2 ([BCIKS20])**.** *Every Reed–Solomon code $\mathcal{C} = \mathsf{RS}[\mathbb{F}, L, d]$ with rate $\rho := (d+1)/|L|$ has a proximity generator for every $\mathsf{m} \in \mathbb{N}$ with proximity bound $\psi := \sqrt{\rho}$, computation time $O(\mathsf{m})$, and one of the following:*

1. *Seed length $\mathsf{w} = \log|\mathbb{F}|$, error $\mathsf{m}\cdot\varepsilon$ or,*
2. *Seed length $\mathsf{w} = \mathsf{m}\cdot\log|\mathbb{F}|$ and error ε*

Above, $\varepsilon := \varepsilon(\delta)$ is defined as follows:

- *If $\delta \in \left(0, \frac{1-\rho}{2}\right]$ then:*

$$\varepsilon(\delta) := \frac{|L|}{|\mathbb{F}|} \ .$$

- *If $\delta \in \left(\frac{1-\rho}{2}, 1-\sqrt{\rho}\right)$ then:*

$$\varepsilon(\delta) := \frac{(d+1)^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{1+\sqrt{\rho}-\delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7} = O\left(\frac{d^2/\rho^4}{|\mathbb{F}|}\right) \ .$$

4.2 Strong proximity generators

We introduce the notion of strong proximity generators. Informally, a strong proximity generator is a generator such that if the linear combination of functions weighted by its output agrees with the Reed–Solomon code *on a specific set*, then this set contains an area where the functions have correlated agreement with the code.

Definition 4.3. *Let $\mathcal{C} \subseteq \mathbb{F}^\ell$ be a code. We say that $\mathcal{C}$ has a* **strong *proximity generator*** *for m functions with seed length w, proximity bound ψ, error ε and computation time t, if there exists a t-time computable function $\mathsf{Gen}\colon \{0,1\}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{m}}$ such that for every $\delta \in (0, 1-\psi)$, functions $f_1 \ldots, f_{\mathsf{m}}\colon [\ell] \to \mathbb{F}$ and every $S \subseteq L$ with $|S| \geq (1-\delta)\cdot|L|$, if*

$$\Pr\left[\begin{array}{l|r} \exists T \subseteq S & s \leftarrow \{0,1\}^{\mathsf{w}} \\ |T| \geq (1-\delta)\cdot|L| \quad : \quad \exists u \in \mathcal{C},\ u(T) = \sum_{j\in[\mathsf{m}]} \xi_j \cdot f_j(T) & (\xi_1,\ldots,\xi_{\mathsf{m}}) := \mathsf{Gen}(s) \end{array}\right] > \varepsilon \ .$$

then there exists a set $W \subseteq S$ with $|W| \geq (1-\delta)\cdot|L|$ where

$$\forall i \in [\mathsf{m}],\ \exists u \in \mathcal{C},\ u(W) = f_i(W) \ .$$

We show that for Reed–Solomon codes, proximity generators imply strong proximity generators with almost the same complexity parameters:

Theorem 4.4. *Let $\mathcal{C} = \mathsf{RS}[\mathbb{F}, L, d]$ be a Reed–Solomon code where $|\mathbb{F}| \geq |L|^2$. Suppose that $\mathcal{C}$ has a proximity generator Gen for $\mathsf{m}+1$ functions with seed length w, proximity bound ψ, error ε, and computation time t.*

Let $\mathsf{Gen}'\colon \{0,1\}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{m}}$ be the restriction of Gen to its first m outputs. Then Gen' is a strong *proximity generator for $\mathcal{C}$ for m functions with seed length w, proximity bound ψ, error ε, and computation time t.*

Theorem 4.4, when instantiated with the proximity generators of Theorem 4.2, implies that every Reed–Solomon code over a large enough field has a strong proximity generator.

Corollary 4.5. *Every Reed–Solomon code $\mathcal{C} = \mathsf{RS}[\mathbb{F}, L, d]$ with $\mathbb{F} \geq |L|^2$ and rate $\rho := (d+1)/|L|$ has a strong proximity generator for every $\mathsf{m} \in \mathbb{N}$ with proximity bound $\psi := \sqrt{\rho}$, computation time $O(\mathsf{m})$, and one of the following:*

1. *Seed length $\mathsf{w} = \log|\mathbb{F}|$ and error $(\mathsf{m}+1)\cdot\varepsilon$ or,*
2. *Seed length $\mathsf{w} = (\mathsf{m}+1)\cdot\log|\mathbb{F}|$ and error ε.*

Above, ε is defined as in Theorem 4.2.

Proof of Theorem 4.4. Let $\delta \in (0,\psi)$ and let $g\colon L \to \mathbb{F}$ be the function described in Claim 4.6 where $g(x) = 0$ for every $x \in S$. Since g is zero on S, for any $\xi_1, \ldots, \xi_{\mathsf{m}+1}$:

$$\xi_{\mathsf{m}+1}\cdot g(S) + \sum_{i\in[\mathsf{m}]} \xi_i \cdot f_i(S) = \sum_{i\in[\mathsf{m}]} \xi_i \cdot f_i(S) \ ,$$

and so

$$\begin{aligned} \varepsilon &< \Pr_{(\xi_1,\ldots,\xi_{\mathsf{m}})\leftarrow\mathsf{Gen}'(U_{\mathsf{w}})}\left[\begin{array}{ll} \exists T \subseteq S & \\ |T| \geq (1-\delta)\cdot|L| & : \quad \exists u \in \mathcal{C},\ u(T) = \sum_{j\in[\mathsf{m}]} \xi_j \cdot f_j(T) \end{array}\right] \\ &= \Pr_{(\xi_1,\ldots,\xi_{\mathsf{m}+1})\leftarrow\mathsf{Gen}(U_{\mathsf{w}})}\left[\begin{array}{ll} \exists T \subseteq S & \\ |T| \geq (1-\delta)\cdot|L| & : \quad \exists u \in \mathcal{C},\ u(T) = \xi_{\mathsf{m}+1}\cdot g(T) + \sum_{j\in[\mathsf{m}]} \xi_j \cdot f_j(T) \end{array}\right] \ , \end{aligned}$$

where U_{w} is the uniform distribution over $\{0,1\}^{\mathsf{w}}$.

Notice that if there exists a subset T of S of size $(1-\delta)\cdot|L|$ such that

$$\exists u \in \mathcal{C},\ u(T) = \xi_{\mathsf{m}+1} \cdot g(T) + \sum_{j\in[\mathsf{m}]} \xi_j \cdot f_j(T)\ ,$$

then, by definition, $\Delta\left(\xi_{\mathsf{m}+1}\cdot g + \sum_{i\in[\mathsf{m}]}\xi_i\cdot f_i, \mathcal{C}\right) \le \delta$. So:

$$\begin{aligned}\varepsilon &< \Pr_{(\xi_1,\ldots,\xi_{\mathsf{m}+1})\leftarrow\mathsf{Gen}(U_{\mathsf{w}})}\left[\begin{array}{ll}\exists T\subseteq S & \\ |T|\ge(1-\delta)\cdot|L| & : \quad \exists u\in\mathcal{C},\ u(T)=\xi_{\mathsf{m}+1}\cdot g(T)+\sum_{j\in[\mathsf{m}]}\xi_j\cdot f_j(T)\end{array}\right] \\ &\le \Pr_{(\xi_1,\ldots,\xi_{\mathsf{m}+1})\leftarrow\mathsf{Gen}(U_{\mathsf{w}})}\left[\Delta\left(\xi_{\mathsf{m}+1}\cdot g+\sum_{i\in[\mathsf{m}]}\xi_i\cdot f_i,\mathcal{C}\right)\le\delta\right]\ .\end{aligned}$$

Since Gen is a proximity generator for m functions with proximity bound ψ and error ε: there exists a set $W\subseteq L$ with $|W|\ge(1-\delta)\cdot|L|$,

$$\forall i\in[\mathsf{m}],\ \exists u\in\mathcal{C},\ f_i(W)=u(W)\ ,$$

and there exists $u\in\mathcal{C}$ with $g(W)=u(W)$.

Since g has $g(S')\neq u(S')$ for every $u\in\mathcal{C}$ and $S'\subseteq L$ where $|S'|\ge(1-\delta)\cdot|L|$ and $S'\neq S$, it follows that $W\subseteq S$. □

Claim 4.6. *Let $\mathcal{C}:=\mathsf{RS}[\mathbb{F},L,d]$ be a Reed–Solomon code with $|\mathbb{F}|\ge|L|^2$. For every S with $|S|\ge(1-\delta)\cdot|L|$ with $\delta\le 1-10d/|L|$ there exists a function $g\colon L\to\mathbb{F}$ where:*

- *$g(x)=0$ for every $x\in S$, and*
- *For every $S'\subseteq L$ where $|S'|\ge(1-\delta)\cdot|L|$ and $S'\neq S$, and every $u\in\mathcal{C}$: $g(S')\neq u(S')$.*

Proof. We show the existence of such a function g via the probabilistic method. Define a distribution G as follows: for every $x\in S$, we set $g(x)=0$ and for every $x\in L\setminus S$, we set $g(x)$ to be a uniformly random element in $\mathbb{F}$.

Fix any set $S'\subseteq L$ with $k:=|S'|\ge(1-\delta)\cdot|L|$, and any nonzero $0\neq u\in\mathcal{C}$. If $|S'\cap S|>d$, then it must be that $g(S')\neq u(S')$. This follows from the fact that u vanishes on $S'\cap S$ which, if $|S'\cap S|>d$ means that $u\equiv 0$.

Thus, assume that $|S'\cap S|\le d$ which means that $|S'\setminus S|\ge|S'|-d\ge 9k/10$ (since $|S'|\ge 10d$ by the requirement on δ). Then, it holds that

$$\Pr_{g\leftarrow G}[g(S')\neq u(S')] = |\mathbb{F}|^{-9k/10}\ .$$

We now take a union bound over all codewords $u\in\mathcal{C}$ and all sets S'. The number of codewords is bounded by $|\mathcal{C}|\le|\mathbb{F}|^{d+1}$. For any $10d\le k\le|L|$, the number of sets of size k is at most $\binom{|L|}{k}\le|L|^k$. Thus,

$$\begin{aligned}&\Pr_{g\leftarrow G}[\exists u\in\mathcal{C}, S'\subseteq L, |S'|=k : g(S')=u(S')] \\ &\le |\mathbb{F}|^{-9k/10}\cdot|\mathbb{F}|^{d+1}\cdot|L|^k \\ &\le |\mathbb{F}|^{-9k/10+d+1+k/2} \\ &= |\mathbb{F}|^{-4k/10+d+1} \\ &= |\mathbb{F}|^{-k/5}\ .\end{aligned}$$

Now, taking values of k over all $10d \leq k \leq |L|$, we get that

$$\begin{aligned}
&\Pr_{g \leftarrow G}[\exists u \in \mathcal{C}, S' \subseteq L, |S'| \geq (1-\delta) \cdot |L| : g(S') = u(S')] \\
&\leq \sum_{k=10d}^{|L|} |\mathbb{F}|^{-k/5} \\
&\leq 1/2 \ .
\end{aligned}$$

Since the probability is less than 1, we know that such a function g exists. □

5 From poly-IOPs to IOPs through low-degree tests

We describe a transformation that combines a poly-IOP and an RS-code IOPP to obtain a corresponding IOP. This transformation works in the high-soundness regime, while similar transformations in prior work provide at best a constant soundness error. Moreover, prior transformations only work with RS-encoded IOPs, a weaker notion compared to poly-IOP. The resulting IOP invokes the given RS-code IOPP only once.

The theorem below is stated for a poly-IOPP (the proximity variant of a poly-IOP) which results in an IOPP. The regular variant follows in a straightforward way.

Theorem 5.1. *Consider the following ingredients:*

- $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$ *is a poly-IOPP for a relation R where the prover sends $\mathsf{m}_{\mathsf{PIOP}}$ polynomials where the i-th polynomial is of degree d_i;*
- $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *is an IOPP for $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{max}}]$ with rate $\rho_{\mathsf{prx}} := (d_{\mathsf{max}}+1)/|L_{\mathsf{prx}}|$;*
- Gen *is proximity generator for $\mathcal{C}_{\mathsf{prx}}$, $2 \cdot \mathsf{m}_{\mathsf{PIOP}}$ functions, seed length $\mathsf{w}_{\mathsf{gen}}$, proximity bound ψ_{gen}, error $\varepsilon_{\mathsf{gen}}$, and computation time $\mathsf{t}_{\mathsf{gen}}$.*

If $\max_{i \in [\mathsf{k}_{\mathsf{PIOP}}]}\{d_i\} \leq d_{\mathsf{max}}$, $0 < \gamma < 1 - \max\{\psi_{\mathsf{gen}}, 2\rho_{\mathsf{prx}}\}$, and $\mathcal{C}_{\mathsf{prx}}$ is (γ, ℓ)–list decodable, then Construction 5.6 is an IOPP $(\mathbf{P}, \mathbf{V})$ for R with the following parameters:[9]

	poly-IOPP for R	IOPP for $\mathcal{C}_{\mathsf{prx}}$	$\to$ IOPP for R
Notation	$(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$	$(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$	$(\mathbf{P}, \mathbf{V})$
Proximity error	β_{PIOP}	β_{prx}	$\max\left\{\beta_{\mathsf{PIOP}}, \beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}} + \mathsf{m}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}|\right\}$
Rounds	$\mathsf{k}_{\mathsf{PIOP}}$	$\mathsf{k}_{\mathsf{prx}}$	$2\mathsf{k}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{prx}} + 1$
Alphabet	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{m}_{\mathsf{PIOP}}$	$\mathsf{l}_{\mathsf{prx}}$	$O(\mathsf{m}_{\mathsf{PIOP}} \cdot \lvert L_{\mathsf{prx}}\rvert + \mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{PIOP},\mathrm{w}}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{PIOP},\mathrm{w}}$
Proof queries	$\mathsf{q}_{\mathsf{PIOP},\pi}$	$\mathsf{q}_{\mathsf{prx},\pi}$	$O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{m}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$
Randomness	$\mathsf{r}_{\mathsf{PIOP}}$	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{PIOP}} \cdot \log\lvert\mathbb{F}\rvert + \mathsf{w}_{\mathsf{gen}} + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\mathsf{vt}_{\mathsf{PIOP}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{PIOP}} + \mathsf{m}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{t}_{\mathsf{gen}} + \mathsf{vt}_{\mathsf{prx}}$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(\gamma)$ and $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$.

This section is organized as follows: (i) in Section 5.1 we discuss univariate function quotienting, a crucial tool in our transformation; (ii) in Section 5.2 we describe our transformation and discuss its efficiency; and (iii) in Section 5.3 we prove its completeness and proximity error.

Remark 5.2. If given an poly-IOPP where the verifier queries only $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}}$ out of the $\mathsf{m}_{\mathsf{PIOP}}$ polynomials, then Theorem 5.1 can be improved: Gen is required to be a proximity generator for only $2 \cdot \mathsf{q}_{\mathsf{PIOP},\mathsf{m}}$ functions (which may lead to changes in its other parameters), and the parameters of the resulting IOPP are:

[9] Note that $\mathsf{m}_{\mathsf{PIOP}}$ counts the number of polynomials sent by the prover, while the proof length for the IOPPs for $\mathcal{C}_{\mathsf{prx}}$ and for R is counted in field elements.

	poly-IOPP for R	*IOPP for* $\mathcal{C}_{\mathsf{prx}}$	$\rightarrow$ *IOPP for* R
Notation	$(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$	$(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$	$(\mathbf{P}, \mathbf{V})$
Proximity error	β_{PIOP}	β_{prx}	$\max\left\{\beta_{\mathsf{PIOP}}, \beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}} + \mathsf{m}_{\mathsf{PIOP}} \cdot d_{\max} \cdot \ell^2/\lvert\mathbb{F}\rvert\right\}$
Rounds	$\mathsf{k}_{\mathsf{PIOP}}$	$\mathsf{k}_{\mathsf{prx}}$	$2\mathsf{k}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{prx}} + 1$
Alphabet	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{m}_{\mathsf{PIOP}}$	$\mathsf{l}_{\mathsf{prx}}$	$O(\mathsf{m}_{\mathsf{PIOP}} \cdot \lvert L_{\mathsf{prx}}\rvert + \mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{PIOP},\mathrm{w}}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{PIOP},\mathrm{w}}$
Proof queries	$\mathsf{q}_{\mathsf{PIOP},\pi}$	$\mathsf{q}_{\mathsf{prx},\pi}$	$O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{q}_{\mathsf{PIOP},\mathrm{m}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$
Randomness	$\mathsf{r}_{\mathsf{PIOP}}$	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{PIOP}} \cdot \log\lvert\mathbb{F}\rvert + \mathsf{w}_{\mathsf{gen}} + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\mathsf{vt}_{\mathsf{PIOP}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{PIOP}} + \mathsf{q}_{\mathsf{PIOP},\mathrm{m}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{t}_{\mathsf{gen}} + \mathsf{vt}_{\mathsf{prx}}$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(\gamma)$ and $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$.

5.1 Univariate function quotienting

We define the quotient and unquotient of a function, the quotient of a polynomial, and prove that quotienting preserves distance.

Definition 5.3. *Let* $f\colon L \to \mathbb{F}$ *be a function,* $S \subseteq L$ *be a set, and* $\mathsf{Ans}, \mathsf{Fill}\colon S \to \mathbb{F}$ *be functions. Let* $\widehat{\mathsf{Ans}} \in \mathbb{F}^{\leq |S|-1}[X]$ *be the (unique) low-degree polynomial extension of* Ans*.*

- *The* **quotient** *function* $\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})\colon L \to \mathbb{F}$ *is defined follows:*

$$\forall\, z \in L\,,\ \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(z) := \begin{cases} \mathsf{Fill}(z) & z \in S \\ \frac{f(z) - \widehat{\mathsf{Ans}}(z)}{\prod_{a \in S}(z-a)} & \textit{otherwise} \end{cases} .$$

- *The* **unquotient** *function* $\mathsf{Unquotient}(f, S, \mathsf{Ans})\colon L \to \mathbb{F}$ *is defined follows:*

$$\forall\, z \in L\,,\ \mathsf{Unquotient}(f, S, \mathsf{Ans})(z) := \widehat{\mathsf{Ans}}(z) + f(z) \cdot \prod_{a \in S}(z - a)\ .$$

Next we define the polynomial quotient operator, which quotients a polynomial relative to its output on evaluation points. This quotient always yields a polynomial of lower degree.

Definition 5.4. *Let* $\hat{f} \in \mathbb{F}^{\leq d}[X]$ *be a polynomial and* $S \subseteq L$ *be a set. Let* $\widehat{\mathsf{Ans}} \in \mathbb{F}^{\leq |S|-1}[X]$ *be the unique polynomial of degree at most* $|S| - 1$ *such that* $\widehat{\mathsf{Ans}}(a) = \hat{f}(a)$ *for every* $a \in S$*. The* **polynomial quotient** $\mathsf{PolyQuotient}(\hat{f}, S) \in \mathbb{F}^{\leq d-|S|}[X]$ *is defined as follows:*

$$\mathsf{PolyQuotient}(\hat{f}, S)(X) := \frac{\hat{f}(X) - \widehat{\mathsf{Ans}}(X)}{\prod_{a \in S}(X - a)}\ .$$

The following claim shows that quotienting preserves distance.

Claim 5.5. *Let* $\mathbb{F}$ *be a field,* $L \subseteq \mathbb{F}$ *be a domain,* $f, u\colon L \to \mathbb{F}$ *be functions,* $S \subseteq \mathbb{F}$ *be a set, and* $\mathsf{Ans}, \mathsf{Fill}\colon S \to \mathbb{F}$ *be functions. Then:*

$$\Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})) \geq \Delta(\mathsf{Unquotient}(u, S, \mathsf{Ans}), f)\ .$$

Proof. Consider the function $w\colon L \to \mathbb{F}$:

$$w := \mathsf{Unquotient}(u, S, \mathsf{Ans}) \ ,$$

and let $Z := \{z \in L \mid u(z) = \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(z)\}$. By definition $\Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})) = 1 - |Z|/|L$. For every $z \in Z$:

$$\begin{aligned} w(z) &= \mathsf{Unquotient}(u, S, \mathsf{Ans})(z) \\ &= \mathsf{Unquotient}(\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}), S, \mathsf{Ans})(z) \\ &= f(z) \ . \end{aligned}$$

We therefore have that $\Delta(f, w) \leq 1 - |Z|/|L| = \Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}))$. □

5.2 Construction

We describe the transformation from a poly-IOPP to an IOPP, and then discuss its efficiency. For simplicity, we assume that the poly-IOPP prover sends a single polynomial in each round, and later discuss how to extend this construction to the case where the poly-IOPP prover sends multiple polynomials in each round.

Construction 5.6. The (honest) IOPP prover $\mathbf{P}$ receives as input an instance-witness pair $(\mathbb{x}, \mathbb{w})$, while the IOPP verifier $\mathbf{V}$ receives as input $\mathbb{x}$ and oracle access to $\mathbb{w}$. They interact as follows.

1. For $i = 1, \ldots, \mathsf{k}_{\mathsf{PIOP}}$:
   (a) $\mathbf{P}$: Compute $\hat{f}_i := \mathbf{P}_{\mathsf{PIOP}}(\mathbb{x}, \mathbb{w}, \rho_1, \ldots, \rho_{i-1}) \in \mathbb{F}^{\leq d_i}[X]$ ($\rho_1, \ldots, \rho_{i-1}$ were received in prior rounds). Compute and send $g_i := \hat{f}_i(L_{\mathsf{prx}}) \in \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_i]$.
   (b) $\mathbf{V}$: Sample and send a random field element $x_i \leftarrow \mathbb{F}$.
   (c) $\mathbf{P}$: Compute and send $y_i := \hat{f}_i(x_i) \in \mathbb{F}$.
   (d) $\mathbf{V}$: Sample and send $\rho_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.

2. $\mathbf{P}$:
   (a) Compute sets $(Q'_i)_{i \in [\mathsf{k}_{\mathsf{PIOP}}]}$ where Q'_i are the queries by $\mathbf{V}_{\mathsf{PIOP}}^{\mathbb{w}, \hat{f}_1, \ldots, \hat{f}_{\mathsf{k}_{\mathsf{PIOP}}}}(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}_{\mathsf{PIOP}}})$ to $\hat{f}_i$.
   (b) For every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, set $Q_i := Q'_i \cup \{x_i\}$.
   (c) For every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, set $\hat{f}_i^Q := \mathsf{PolyQuotient}(\hat{f}_i, Q_i) \in \mathbb{F}^{\leq d_i - |Q_i|}$.
   (d) Send $((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i \in [\mathsf{k}_{\mathsf{PIOP}}]}$ where $\mathsf{Ans}_i\colon Q_i \to \mathbb{F}$ is $\hat{f}_i(Q_i)$, and $\mathsf{Fill}_i\colon Q_i \to \mathbb{F}$ is $\hat{f}_i^Q(Q_i)$.

3. $\mathbf{V}$:
   (a) For every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, set the degree $\sigma_i := d_{\mathsf{max}} - (d_i - |Q_i|)$.
   (b) Sample and send $s \leftarrow \{0,1\}^{\mathsf{w}_{\mathsf{gen}}}$. Set $(\xi_1, \ldots, \xi_{2\mathsf{k}_{\mathsf{PIOP}}}) := \mathsf{Gen}(s)$.

4. For every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, set $h_i := \mathsf{Quotient}\,(g_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i)$. This defines a function $u\colon L_{\mathsf{prx}} \to \mathbb{F}$:

$$\forall\, x \in L_{\mathsf{prx}}\,,\ u(x) := \sum_{i \in [\mathsf{k}_{\mathsf{PIOP}}]} \xi_i \cdot h_i(x) + \xi_{2i} \cdot x^{\sigma_i} \cdot h_i(x) \ .$$

   The parties run the interaction phase of the IOPP $\langle \mathbf{P}_{\mathsf{prx}}(\mathcal{C}_{\mathsf{prx}}, \hat{u}), \mathbf{V}^u_{\mathsf{prx}}(\mathcal{C}_{\mathsf{prx}}) \rangle$.

5. $\mathbf{V}$ accepts if and only if the following checks pass.

(a) $\mathbf{V}_{\mathsf{PIOP}}^{\mathbb{w},\hat{f}_1,\ldots,\hat{f}_{\mathsf{k}_{\mathsf{PIOP}}}}(\mathbb{x},\rho_1,\ldots,\rho_{\mathsf{k}_{\mathsf{PIOP}}}) = 1$, where a query a to $\hat{f}_i$ is answered by $b := \mathsf{Ans}_i(a)$ (reject if $a \notin Q_i$) and queries to $\mathbb{w}$ are answered by querying $\mathbb{w}$ directly.
(b) For every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, $\mathsf{Ans}_i(x_i) = y_i$.
(c) $\mathbf{V}_{\mathsf{prx}}$ accepts in its decision phase, when $\mathbf{V}$ answers a query t made by $\mathbf{V}_{\mathsf{prx}}$ to u as follows:
  i. for every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, query g_i at t;
  ii. for every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, compute $\ell_i := \mathsf{Quotient}\left(g_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i\right)(t)$;
  iii. return the value $u(t) := \sum_{i\in[\mathsf{k}_{\mathsf{PIOP}}]} \xi_i \cdot \ell_i + \xi_{2i} \cdot t^{\sigma_i} \cdot \ell_i$.

Remark 5.7. If the poly-IOPP prover sends multiple polynomials in each round, then the above protocol is altered in the following ways:
- The proximity generator Gen is required to work for $2 \cdot \mathsf{m}_{\mathsf{PIOP}}$ functions. As a result, the coefficients generated by it are indexed until $2\mathsf{m}_{\mathsf{PIOP}}$, rather than $2\mathsf{k}_{\mathsf{PIOP}}$.
- In Item 1a, the prover sends a separate function g for each message sent.
- In Item 1c, the prover sends answers y separately for each message sent (notice that there is only one element x_i per round, even if multiple polynomials are sent).
- In Item 4, and in the remaining protocol, the function u is the weighted sum of all $\mathsf{m}_{\mathsf{PIOP}}$ functions sent by the prover (and their degree corrections).

Complexity parameters. We analyze the complexity parameters of the new IOPP.

- *Rounds.* The new IOPP begins by emulating the $\mathsf{k}_{\mathsf{PIOP}}$-round poly-IOPP and, for each such round, the IOPP has two rounds of interaction (each poly-IOPP round is followed by an out-of-domain sampling round). Next, the verifier sends the seed for Gen, and both parties run the $\mathsf{k}_{\mathsf{prx}}$-round IOPP for $\mathcal{C}_{\mathsf{prx}}$. This results in a total of $2\mathsf{k}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{prx}} + 1$ rounds.

- *Proof length.* For every polynomial sent by the poly-IOPP prover, the IOPP prover sends $|L_{\mathsf{prx}}| + 1$ field elements. Next, the IOPP prover sends $O(\mathsf{q}_{\mathsf{PIOP},\pi})$ field elements to represent the list $((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}$. Finally, the IOPP prover emulates $\mathbf{P}_{\mathsf{prx}}$. Thus, the overall proof length is $O(\mathsf{k}_{\mathsf{PIOP}} \cdot |L_{\mathsf{prx}}| + \mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{l}_{\mathsf{prx}}$. (This changes to $O(\mathsf{m}_{\mathsf{PIOP}} \cdot |L_{\mathsf{prx}}| + \mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{l}_{\mathsf{prx}}$ if the poly-IOPP prover sends multiple polynomials per round, and $\mathsf{m}_{\mathsf{PIOP}}$ polynomials in total.)

- *Oracle input queries.* The new IOPP verifier queries $\mathbb{w}$ only when $\mathbf{V}_{\mathsf{PIOP}}$ queries $\mathbb{w}$. Hence the number of oracle input queries is $\mathsf{q}_{\mathsf{PIOP},\mathbb{w}}$.

- *Proof queries.* The new IOPP verifier queries $(y_i)_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}$ and the list $((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}$. Moreover, each oracle input query of the low-degree test yields $\mathsf{k}_{\mathsf{PIOP}}$ queries, and each proof query remains one proof query. The query complexity is therefore $O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{k}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$. (This value changes to $O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{m}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$ if the poly-IOPP prover sends multiple polynomials per round, and $\mathsf{m}_{\mathsf{PIOP}}$ polynomials in total.)

- *Randomness.* The new IOPP verifier uses at most $\mathsf{r}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{PIOP}} \cdot \log|\mathbb{F}| + \mathsf{w}_{\mathsf{gen}} + \mathsf{r}_{\mathsf{prx}}$ bits of randomness. (This value does not change if the poly-IOPP prover sends multiple polynomials per round, as the the same random point x_i be used for every message sent in the same round without affecting the proximity error.)

- *Verifier running time.* The new IOPP verifier runs the poly-IOPP verifier where for every polynomial sent by the prover it sends a field element, it runs the proximity generator, and the low-degree test verifier. Additionally, it must interpolate Ans_i in time $\tilde{O}(|Q_i|)$ order to compute

the quotients. The verifier therefore runs in time $O(\mathsf{vt}_{\mathsf{PIOP}} + \mathsf{k}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{t}_{\mathsf{gen}} + \mathsf{vt}_{\mathsf{prx}}$. (This value changes to $O(\mathsf{vt}_{\mathsf{PIOP}} + \mathsf{m}_{\mathsf{PIOP}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\pi}) + \mathsf{t}_{\mathsf{gen}} + \mathsf{vt}_{\mathsf{prx}}$ if the poly-IOPP prover sends multiple polynomials in a single round since the verifier must query every function in the proximity test.)

5.3 Completeness and soundness

We prove Theorem 5.1. We first analyze completeness and then soundness.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in R$. We show that $\Pr[\langle \mathbf{P}(\mathbb{x}, \mathbb{w}), \mathbf{V}^{\mathbb{w}}(\mathbb{x})\rangle] = 1$. Fix any transcript of this protocol, which has the following structure:

$$\mathsf{tr} = \left(((g_i, x_i, y_i, \rho_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}, ((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}, s, \mathsf{tr}_{\mathsf{prx}}\right) \ ,$$

where $\mathsf{tr}_{\mathsf{prx}}$ is a transcript of $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$. Let $(\xi_1, \ldots, \xi_{2\mathsf{k}_{\mathsf{PIOP}}}) := \mathsf{Gen}(s)$.

Perfect completeness of $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$ implies that

$$\mathbf{V}_{\mathsf{PIOP}}^{\mathbb{w},\hat{f}_1,\ldots,\hat{f}_{\mathsf{k}_{\mathsf{PIOP}}}}(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}_{\mathsf{PIOP}}}) = 1$$

where $\hat{f}_i = \mathbf{P}_{\mathsf{PIOP}}(\mathbb{x}, \mathbb{w}, \rho_1, \ldots, \rho_i)$ is a polynomial of degree (at most) d_i; moreover, in this execution $\mathbf{V}_{\mathsf{PIOP}}$ queries $\hat{f}_i$ at a set $Q'_i \subseteq Q_i$.

Since $\mathbf{P}$ sends $\mathsf{Ans}_i = f_i(Q_i)$, $\mathbf{V}$ answers every query of $\mathbf{V}_{\mathsf{PIOP}}$ consistently with the polynomials $\hat{f}_1, \ldots, \hat{f}_{\mathsf{k}_{\mathsf{PIOP}}}$, which means that $\mathbf{V}$ does not reject in Item 5a. Moreover, $\mathbf{V}$ does not reject in Item 5b either, since $\mathsf{Ans}_i(x_i) = y_i$ for each i. We are left to argue that $\mathbf{V}$ does not reject in Item 5c, that is, $\mathbf{V}_{\mathsf{prx}}$ accepts.

Observe that, by definition,

$$\hat{f}_i^Q(X) := \mathsf{PolyQuotient}(\hat{f}_i, Q_i)(X) = \frac{\hat{f}_i(X) - \widehat{\mathsf{Ans}}_i(X)}{\prod_{a\in Q_i}(X-a)} \ ,$$

and has degree at most $d_i - |Q_i|$. Furthermore, observe that $\mathsf{Fill}_i(t) = \hat{f}_i^Q(t)$ for every $t \in Q_i$.

Let h_i be as in the protocol description and let $\hat{h}_i$ be the interpolation of h_i to a polynomial. We argue that $\hat{f}_i^Q \equiv \hat{h}$ by showing that they agree at every location, and in particular $\hat{h}$ has degree at most $d_i - |Q_i|$. Observe that for every $t \in \mathbb{F}$:

$$\hat{h}_i(t) := \mathsf{Quotient}(\hat{f}_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i)(t) = \begin{cases} \mathsf{Fill}_i(t) & t \in Q_i \\ \frac{f(t) - \widehat{\mathsf{Ans}}_i(t)}{\prod_{a\in Q_i}(t-a)} & \text{otherwise} \end{cases} \ .$$

Hence:

- For every $t \in Q_i$, $\hat{h}_i(t) = \mathsf{Fill}_i(t) = \hat{f}_i^Q(t)$.
- For every $t \notin Q_i$, $\hat{h}(t)$ and $\hat{f}_i^Q(t)$ are defined identically and are therefore equal.

It follows that $\hat{h}_i$ and $\hat{f}_i^Q$ are identical.

Since $\sigma_i := d_{\mathsf{max}} - (d_i - |Q_i|)$, both $\hat{h}_i$ and $X^{\sigma_i} \cdot \hat{h}_i$ have degree at most d_{max}. Consequently, for any coefficients $\xi_1, \ldots, \xi_{2\mathsf{k}_{\mathsf{PIOP}}}$, the following linear combination has degree at most d_{max}:

$$\hat{u}(X) := \sum_{j\in[\mathsf{k}_{\mathsf{PIOP}}]} \xi_i \cdot \hat{h}_i(X) + \xi_{2i} \cdot X^{\sigma_i} \cdot \hat{h}_i(X) \ .$$

Hence $u := \hat{u}(L_{\mathsf{prx}}) \in \mathcal{C}_{\mathsf{prx}}$. By the perfect completeness of $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$,

$$\Pr\left[\langle \mathbf{P}_{\mathsf{prx}}(\mathcal{C}_{\mathsf{prx}}, \hat{u}), \mathbf{V}^{u}_{\mathsf{prx}}(\mathcal{C}_{\mathsf{prx}})\rangle = 1\right] = 1 \ .$$

Finally, $\mathbf{V}$ gives $\mathbf{V}_{\mathsf{prx}}$ virtual oracle access to u, so $\mathbf{V}$ accepts with probability 1.

Soundness. Fix $\mathtt{x} \notin L(R)$ and a malicious prover $\tilde{\mathbf{P}}$ for the IOPP $(\mathbf{P}, \mathbf{V})$. A transcript of the protocol has the following structure:

$$\mathsf{tr} = \left(((g_i, x_i, y_i, \rho_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}, ((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{PIOP}}]}, s, \mathsf{tr}_{\mathsf{prx}}\right) \ .$$

Let $(\xi_1, \ldots, \xi_{2\mathsf{k}_{\mathsf{PIOP}}}) := \mathsf{Gen}(s)$ and, for every $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, let $h_i := \mathsf{Quotient}(g_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i)$. Define the following events:

- $\mathsf{E}_{\mathsf{out}}$ is the event that for every i there is at most one polynomial $\hat{w}_i \in \mathbb{F}^{\leq d_i}[X]$ with $\Delta(g_i, \hat{w}_i(L_{\mathsf{prx}})) \leq \gamma$ and $\hat{w}_i(x_i) = y_i$; and
- $\mathsf{E}_{\mathsf{prx}}$ is the event that for every i there exists at least one polynomial $\hat{v}_i \in \mathbb{F}^{\leq d_i - |Q_i|}[X]$ with $\Delta(h_i, \hat{v}_i(L_{\mathsf{prx}})) \leq \gamma$.

The following claim shows that, conditioned on $\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}$, the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}$ is bounded by the soundness error of $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$.

Claim 5.8. $\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathrm{w}}\rangle(\mathtt{x}) = 1 \mid \mathsf{E}_{\mathsf{prx}} \ \wedge \ \mathsf{E}_{\mathsf{out}}\ ] \leq \beta_{\mathsf{PIOP}}$.

Proof. Suppose towards contradiction that the above probability is greater than β_{PIOP}. Below we construct a prover $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ that convinces $\mathbf{V}_{\mathsf{PIOP}}$ on $\mathtt{x}$ with probability greater than β_{PIOP}, contradicting the soundness property of $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$. Prior to round $i \in [\mathsf{k}_{\mathsf{PIOP}}]$, $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ has chosen (and stored) points $x_1, \ldots, x_{i-1}$ and has received the messages $\rho_1, \ldots, \rho_{i-1}$ from the verifier.

$\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ in round $i \in [\mathsf{k}_{\mathsf{PIOP}}]$:

1. Compute $g_i := \tilde{\mathbf{P}}(x_1, \rho_1, \ldots, x_{i-1}, \rho_{i-1})$.
2. Sample $x_i \leftarrow \mathbb{F}$. (Pass this value on as state for the next round.)
3. Compute $y_i := \tilde{\mathbf{P}}(x_1, \rho_1, \ldots, x_{i-1}, \rho_{i-1}, x_i)$. If there is exactly one polynomial $\hat{w}_i \in \mathbb{F}^{\leq d_i}[X]$ such that $\hat{w}_i(x_i) = y_i$ and $\Delta(\hat{w}_i(L_{\mathsf{prx}}), g_i) \leq \gamma$ then send $\hat{w}_i$. Otherwise abort.
4. Receive ρ_i from $\mathbf{V}_{\mathsf{PIOP}}$.

We analyze the probability that $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ convinces $\mathbf{V}_{\mathsf{PIOP}}$. We show that this probability is bounded from below by the probability that $\mathbf{V}$ accepts conditioned on $\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}$.

Since $\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}$ holds, for every i there exists a polynomial $\hat{v}_i \in \mathbb{F}^{\leq d_i - |Q_i|}[X]$ where $\Delta(\hat{v}_i(L_{\mathsf{prx}}), h_i) \leq \gamma$. Consider the polynomial $\hat{w}_i = \mathsf{Unquotient}(\hat{v}_i, Q_i, \mathsf{Ans}_i) \in \mathbb{F}^{\leq d_i}[X]$. Since $h_i := \mathsf{Quotient}(g_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i)$, by Claim 5.5: $\Delta(\hat{w}_i(L_{\mathsf{prx}}), g_i) \leq \Delta(\hat{v}_i(L_{\mathsf{prx}}), h_i) \leq \gamma$.

Observe that $\hat{w}_i(a) = \mathsf{Ans}_i(a)$ for every $a \in Q_i$. In particular, $\hat{w}_i(x_i) = y_i$. Since $\mathsf{E}_{\mathsf{out}}$ holds, $\hat{w}_i$ is unique, and therefore $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ sends $\hat{w}_i$ to $\mathbf{V}_{\mathsf{PIOP}}$.

If $\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}$ holds and $\mathbf{V}$ accepts, it must be that $\mathbf{V}_{\mathsf{PIOP}}$ makes the queries in $Q_1, \ldots, Q_{\mathsf{k}_{\mathsf{PIOP}}}$ and accepts the query answers in $\mathsf{Ans}_1, \ldots, \mathsf{Ans}_{\mathsf{k}_{\mathsf{PIOP}}}$. For each i, the polynomial $\hat{w}_i$ chosen by $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$ agrees with these query answers. Consequently,

$$\Pr[\ \langle \tilde{\mathbf{P}}_{\mathsf{PIOP}}, \mathbf{V}^{\mathrm{w}}_{\mathsf{PIOP}}\rangle(\mathtt{x}) = 1\ ] \geq \Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathrm{w}}\rangle(\mathtt{x}) = 1 \mid \mathsf{E}_{\mathsf{prx}} \ \wedge \ \mathsf{E}_{\mathsf{out}}\ ] > \beta_{\mathsf{IOP}} \ ,$$

in contradiction to the bound β_{PIOP} on the soundness of $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$. □

Next we bound the probability that there exist multiple codewords consistent with the out-of-domain samples.

Claim 5.9. $\Pr[\,\neg\mathsf{E}_{\mathsf{out}}\,] \leq \mathsf{k}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}|$. *(If the prover sends $\mathsf{m}_{\mathsf{PIOP}}$ messages overall, then this is amended to:* $\Pr[\,\neg\mathsf{E}_{\mathsf{out}}\,] \leq \mathsf{m}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}|$*.)*

Proof. For every i, $\mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_i]$ is (γ, ℓ)-list decodable: $\mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_i] \subseteq \mathcal{C}_{\mathsf{prx}}$ and $\mathcal{C}_{\mathsf{prx}}$ is (γ, ℓ)-list decodable (see Fact 3.4).

Fix $i \in [\mathsf{k}_{\mathsf{PIOP}}]$ and consider $g_i \colon L_{\mathsf{prx}} \to \mathbb{F}$ sent by $\tilde{\mathbf{P}}_{\mathsf{PIOP}}$. By the polynomial identity lemma (Lemma 3.8), two distinct of polynomials $\hat{u}, \hat{w} \in \mathbb{F}^{\leq d_i}[X]$ agree on at most d_i points of $\mathbb{F}$. There are at most $\binom{\ell}{2}$ pairs of polynomials that are γ-close to g_i on L_{prx}. It follows that the probability over the choice of $x_i \leftarrow \mathbb{F}$ that there exist two distinct polynomials that are γ-close to g_i and agree on x_i is at most $\frac{d_i \cdot \binom{\ell}{2}}{|\mathbb{F}|} \leq \frac{d_{\mathsf{max}} \cdot \ell^2}{|\mathbb{F}|}$.

The claim follows by a union-bound over each index $i \in [\mathsf{k}_{\mathsf{PIOP}}]$. □

Next we bound the probability that the prover $\tilde{\mathbf{P}}$ convinces $\mathbf{V}$ when $\mathsf{E}_{\mathsf{prx}}$ does not occur.

Claim 5.10. $\Pr[\,\langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathrm{w}} \rangle(\mathbb{x}) = 1 \mid \neg\mathsf{E}_{\mathsf{prx}}\,] \leq \varepsilon_{\mathsf{gen}}(\gamma) + \beta_{\mathsf{prx}}$.

Proof. Since $\mathsf{E}_{\mathsf{prx}}$ does not hold, there exists $i \in [\mathsf{k}_{\mathsf{PIOP}}]$ such that every polynomial $\hat{v}_i \in \mathbb{F}^{\leq d_i - |Q_i|}[X]$ has $\Delta(\hat{v}_i(L_{\mathsf{prx}}), h_i) > \gamma$. We will argue that in this case

$$\Pr[\,\Delta(u, \mathcal{C}_{\mathsf{prx}}) \leq \gamma\,] \leq \varepsilon_{\mathsf{gen}}(\gamma)\ . \tag{1}$$

If $\Delta(u, \mathcal{C}_{\mathsf{prx}}) > \gamma$, then by soundness of $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$, $\mathbf{V}_{\mathsf{prx}}$ accepts u with probability at most β_{prx}. Since $\mathbf{V}$ accepts only if $\mathbf{V}_{\mathsf{prx}}$ accepts, it follows that $\mathbf{V}$ accepts with probability at most $\varepsilon_{\mathsf{gen}}(\gamma) + \beta_{\mathsf{prx}}$. We now show that Equation 1 holds.

Suppose towards contradiction that

$$\Pr[\,\Delta(u, \mathcal{C}_{\mathsf{prx}}) \leq \gamma] > \varepsilon_{\mathsf{gen}}(\gamma)\ . \tag{2}$$

Recall that

$$u(x) := \sum_{j \in [\mathsf{k}_{\mathsf{PIOP}}]} \xi_i \cdot h_i(x) + \xi_{2i} \cdot x^{\sigma_i} \cdot h_i(x)\ ,$$

where $(\xi_1, \ldots, \xi_{2\mathsf{k}_{\mathsf{PIOP}}}) := \mathsf{Gen}(s)$ for uniformly random seed s. Observe that $0 < \gamma < 1 - \psi_{\mathsf{gen}}$, and that Gen is a proximity generator for $2 \cdot \mathsf{k}_{\mathsf{PIOP}}$ with proximity bound ψ_{gen} and error $\varepsilon_{\mathsf{gen}}$. Therefore, as a result of Equation 2, it holds that h_i and $X^{\sigma_i} \cdot h_i$ have correlated agreement with the code $\mathcal{C}_{\mathsf{prx}}$ on a set S of size $|S| \geq (1-\gamma) \cdot |L_{\mathsf{prx}}|$. Let p and q be the closest codewords that agree with h_i and $X^{\sigma_i} \cdot h_i$ on S respectively and let $\hat{p}, \hat{q} \in \mathbb{F}^{\leq d_{\mathsf{max}}}[X]$ be the polynomials that agree with p and q. Additionally, let $\hat{p}'_i(X) := X^{\sigma_i} \cdot \hat{p}(X)$.

We now reach a contradiction by showing that it simultaneously holds that $\Delta(\hat{p}'_i(L_{\mathsf{prx}}), \hat{q}(L_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$ and that $\Delta(\hat{p}'_i(L_{\mathsf{prx}}), \hat{q}(L_{\mathsf{prx}})) < 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$:

- $\Delta(\hat{p}'_i(L_{\mathsf{prx}}), \hat{q}(L_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$: since every polynomial $\hat{v}_i \in \mathbb{F}^{\leq d_i - |Q_i|}[X]$ has $\Delta(\hat{v}_i(L_{\mathsf{prx}}), h_i) > \gamma$, and $\Delta(\hat{p}(L_{\mathsf{prx}}), h_i) \leq \gamma$, it follows that $d_i - |Q_i| < \mathsf{deg}(\hat{p}) \leq d_{\mathsf{max}}$. Letting $\hat{p}'_i(X) := X^{\sigma_i} \cdot \hat{p}(X)$, we have $d_{\mathsf{max}} < \mathsf{deg}(\hat{p}') \leq 2d_{\mathsf{max}}$. Since $\hat{q}$ has degree at most d_{max} it holds that $\hat{p}' \not\equiv \hat{q}$. It follows that the two polynomials can agree on at most $2d_{\mathsf{max}}$ points, and so $\Delta(\hat{p}'(L_{\mathsf{prx}}), \hat{q}(L_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$.

- $\Delta(\hat{p}'_i(L_{\mathsf{prx}}), \hat{q}(L_{\mathsf{prx}})) < 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$: This follows from the observation that $X^{\sigma_i} \cdot \hat{p}$ and $\hat{q}$ agree on S, where $|S| \geq (1-\gamma) \cdot |L_{\mathsf{prx}}|$ for $\gamma < 1 - 2\rho_{\mathsf{prx}} < 1 - 2d_{\mathsf{max}}/|L_{\mathsf{prx}}|$.

As we reached a contradiction, we can conclude that Equation 1 holds, thereby proving the claim. $\square$

Define $p := \Pr[\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}]$. Putting together Claim 5.8, Claim 5.9, and Claim 5.10, we obtain:

$$\begin{aligned}
\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1\ ] &= p \cdot \Pr[\ \langle \mathbf{P}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1 \mid \mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}\ ] + (1-p) \cdot \Pr[\ \langle \mathbf{P}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1 \mid \neg\mathsf{E}_{\mathsf{prx}} \vee \neg\mathsf{E}_{\mathsf{out}}\ ] \\
&\leq p \cdot \beta_{\mathsf{PIOP}} + (1-p) \cdot \Big(\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1 \mid \neg\mathsf{E}_{\mathsf{prx}}\ ] + \Pr[\ \neg\mathsf{E}_{\mathsf{out}}\ ] \Big) \\
&\leq p \cdot \beta_{\mathsf{PIOP}} + (1-p) \cdot \big(\beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}}(\gamma) + \mathsf{k}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}| \big) \\
&\leq \max\left\{ \beta_{\mathsf{PIOP}}, \beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}}(\gamma) + \mathsf{k}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}| \right\} \ .
\end{aligned}$$

If the prover sends $\mathsf{m}_{\mathsf{PIOP}}$ messages overall, then this is amended to:

$$\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1\ ] \leq \max\left\{ \beta_{\mathsf{PIOP}}, \beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}}(\gamma) + \mathsf{m}_{\mathsf{PIOP}} \cdot d_{\mathsf{max}} \cdot \ell^2/|\mathbb{F}| \right\} \ .$$

6 High-soundness small-query test for RS codes

We construct an IOPP for RS codes that has small soundness error and small query complexity.

Theorem 6.1. *Let $\mathbb{F}$ be a field, G be a multiplicative subgroup of $\mathbb{F}^*$ whose order is a power of two, and $L \subseteq \mathbb{F}$ be a set.*

If $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot |L|$ then the Reed–Solomon code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ has an IOPP with the following parameters:

IOPP to show δ proximity to $\mathcal{C}$	
Proximity error	$\max\left\{1-\delta,\ \rho^{1/4} + O\left(\frac{d^2\cdot(1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)\right\}$
Rounds	$O(\log\log d)$
Alphabet	$\mathbb{F}$
Proof length	$O(\lvert L\rvert/\rho)$
Oracle input queries	1
Proof queries	$O(\log\log d)$
Randomness	$O(\log\log d \cdot \log\lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(\sqrt{d})$

Above, $\rho := (d+1)/|L|$ is the rate of $\mathcal{C}$.

This section is organized in four parts.

- In Section 6.1 we describe a univariate sumcheck that we use as a subroutine.
- In Section 6.2 we construct a poly-IOPP for bivariate Reed–Muller codes.
- In Section 6.3 we use the above test to construct a poly-IOPP for Reed–Solomon codes of degree d where prover messages have degree $\sqrt{d}$.
- In Section 6.4 we use the compiler in Section 5 to recursively transform the poly-IOPP from Section 6.3 into a (standard) IOPP. This recursion uses the fact that the poly-IOPP has prover messages of degree $\sqrt{d}$ to go from degree d to degree $\sqrt{d}$ and so on.

6.1 Weighted univariate sumcheck

We describe a poly-IOP for univariate sumcheck with weights. This protocol is a straightforward variation of the univariate sumcheck in [BCRSVW19], and is described here for completeness.

Lemma 6.2. *Let H be a multiplicative subgroup of $\mathbb{F}^*$. Let $\hat{f} \in \mathbb{F}^{\leq d_f}[X]$ be a polynomial, $\hat{w} \in \mathbb{F}^{\leq d_w}[X]$ be a weight polynomial, and σ be a claimed sum. The protocol $(\mathbf{P}_\Sigma, \mathbf{V}_\Sigma)$ in Construction 6.3 satisfies the following properties.*

- ***Completeness.*** *If $\sum_{\alpha\in H} \hat{w}(\alpha)\cdot\hat{f}(\alpha) = \sigma$ then*

$$\Pr_{a\leftarrow\mathbb{F}}\left[\begin{array}{l|l} \hat{p}\in\mathbb{F}^{\leq|H|-2}[X] & \\ \wedge\ \hat{h}\in\mathbb{F}^{\leq d_f+d_w-|H|}[X] & (\hat{p},\hat{h}) \leftarrow \mathbf{P}_\Sigma(d_f,H,\hat{w},\sigma,\hat{f}) \\ \wedge\ \mathbf{V}_\Sigma^{\hat{f},\hat{p},\hat{h}}(d_f,H,\hat{w},\sigma,a)=1 & \end{array}\right] = 1 \ .$$

- ***Soundness.*** *If $\sum_{\alpha\in H} \hat{w}(\alpha)\cdot\hat{f}(\alpha) \neq \sigma$ then for every $\tilde{\mathbf{P}}$:*

$$\Pr_{a\leftarrow\mathbb{F}}\left[\begin{array}{l|l} \hat{p}\in\mathbb{F}^{\leq|H|-2}[X] & \\ \wedge\ \hat{h}\in\mathbb{F}^{\leq d_f+d_w-|H|}[X] & (\hat{p},\hat{h}) \leftarrow \tilde{\mathbf{P}} \\ \wedge\ \mathbf{V}_\Sigma^{\hat{f},\hat{p},\hat{h}}(d_f,H,\hat{w},\sigma,a)=1 & \end{array}\right] \leq \frac{d_f+d_w}{|\mathbb{F}|} \ .$$

The protocol has 1 *message, where the prover sends* 2 *polynomials. The verifier queries* 1 *field element from* $\hat{f}$ *and* 2 *from the prover messages, uses* $\log|\mathbb{F}|$ *bits of randomness, and runs in time* $O(\log|H|) + \mathtt{t}_w$ *(field operations) where* $\mathtt{t}_w$ *is the time to evaluate* $\hat{w}$ *on a random field element.*

Construction 6.3.

1. *Interaction phase:* $\mathbf{P}_\Sigma$ sends the polynomials $\hat{h} \in \mathbb{F}^{\leq d_f + d_w - |H|}[X]$ and $\hat{p} \in \mathbb{F}^{\leq |H|-2}[X]$ such that

$$\hat{w}(X) \cdot \hat{f}(X) \equiv \hat{h}(X) \cdot \hat{V}_H(X) + (X \cdot \hat{p}(X) + \sigma/|H|) \ ,$$

   where $\hat{V}_H(X) := \prod_{\alpha \in H}(X - \alpha) = X^{|H|} - 1$ is the vanishing polynomial of H.

2. *Decision phase:* $\mathbf{V}_\Sigma$ samples $a \leftarrow \mathbb{F}$ uniformly at random and accepts if and only if

$$\hat{w}(a) \cdot \hat{f}(a) = \hat{h}(a) \cdot \hat{V}_H(a) + (a \cdot \hat{p}(a) + \sigma/|H|) \ .$$

Proof of Lemma 6.2. We prove completeness and then soundness. We rely on the following fact.

Fact 6.4. *Let* H *be a multiplicative subgroup of* $\mathbb{F}^*$ *and* $\hat{q} \in \mathbb{F}^{\leq |H|-1}[X]$ *be a polynomial. Then* $\sum_{\alpha \in H} \hat{q}(\alpha) = \hat{q}(0) \cdot |H|$.

Completeness. Suppose that $\sum_{\alpha \in H} \hat{w}(\alpha) \cdot \hat{f}(\alpha) = \sigma$. By polynomial division we can write

$$\hat{w}(X) \cdot \hat{f}(X) = \hat{h}(X) \cdot \hat{V}_H(X) + \hat{q}(X) \ ,$$

where $\hat{h} \in \mathbb{F}^{\leq d_f + d_w - |H|}$ and $\hat{q} \in \mathbb{F}^{\leq |H|-1}$. By Fact 6.4, $\sum_{\alpha \in H} \hat{q}(\alpha) = \hat{q}(0) \cdot |H|$. Since $\sigma = \sum_{\alpha \in H} \hat{w}(\alpha) \cdot \hat{f}(\alpha) = \sum_{\alpha \in H} \hat{q}(\alpha)$, we can write:

$$\hat{w}(X) \cdot \hat{f}(X) = \hat{h}(X) \cdot \hat{V}_H(X) + (X \cdot \hat{p}(X) + \sigma/|H|) \ ,$$

for $\hat{p} \in \mathbb{F}^{\leq |H|-2}[X]$. This is precisely what is sent by $\mathbf{P}_\Sigma$. Thus, for every $a \in \mathbb{F}$ it holds that

$$\hat{w}(a) \cdot f(a) = h(a) \cdot \hat{V}_H(a) + (a \cdot p(a) + \sigma/|H|) \ .$$

Thus, $\mathbf{V}_\Sigma$ always accepts.

Soundness. Suppose that $\sum_{\alpha \in H} \hat{w}(\alpha) \cdot \hat{f}(\alpha) \neq \sigma$ and fix a prover $\tilde{\mathbf{P}}$. Let $\hat{h} \in \mathbb{F}^{\leq d_f + d_w - |H|}[X]$ and $\hat{p} \in \mathbb{F}^{\leq |H|-2}[X]$ be the polynomials sent by $\tilde{\mathbf{P}}$. Since the sum does not hold:

$$\hat{w}(X) \cdot \hat{f}(X) \not\equiv \hat{h}(X) \cdot \hat{V}_H(X) + (X \cdot \hat{p} + \sigma/|H|) \ .$$

(This holds since, by using Fact 6.4, otherwise $\sum_{\alpha \in H} \hat{w}(\alpha) \cdot \hat{f}(\alpha) = \sigma$.) Since both polynomials have degree at most $d_f + d_w$, by the polynomial identity lemma (Lemma 3.8):

$$\hat{w}(a) \cdot \hat{f}(a) = \hat{h}(a) \cdot \hat{V}_H(a) + (a \cdot \hat{p}(a) + \sigma/|H|) \ ,$$

can hold for at most $d_f + d_w$ choices of $a \in \mathbb{F}$. Hence $\mathbf{V}_\Sigma$ accepts with probability at most $\frac{d_f + d_w}{|\mathbb{F}|}$. □

6.2 poly-IOPP for bivariate RM codes

We construct a poly-IOPP for bivariate Reed–Muller codes.

Definition 6.5. *A domain $D \subseteq \mathbb{F} \times \mathbb{F}$ is* **admissible** *if there exists a "row-index" domain $L_X \subseteq \mathbb{F}$ such that $D = \cup_{i \in L_X}(\{i\} \times L_Y^{(i)})$ where $|L_Y^{(i)}| = |L_Y^{(i')}|$ for every $i, i' \in L_X$.*

Lemma 6.6. *Consider the following ingredients:*

- $\mathcal{C} := \mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$ *is a Reed–Muller code where $D \subseteq \mathbb{F} \times \mathbb{F}$ is an admissible domain with row-index domain L_X.*
- $H \subseteq \mathbb{F}$ *is a multiplicative subgroup of $\mathbb{F}^*$.*
- Gen *is a strong proximity generator for $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, d_X]$ for $|H|$ functions with seed length $\mathsf{w}_{\mathsf{gen}}$, proximity bound ψ_{gen}, and error $\varepsilon_{\mathsf{gen}}$.*

If $|\mathbb{F}| \geq |L_X|^2$ and $|H| \geq d_Y$, then Construction 6.7 is a poly-IOPP for $\mathcal{C}$ with the following parameters:

poly-IOPP to show $\delta < 1 - \psi_{\mathsf{gen}}^2$ proximity to $\mathcal{C}$	
Proximity error	$\sqrt{1-\delta} + \max\left\{\sqrt{1-\delta},\ \frac{d_Y + \lvert H\rvert - 1}{\lvert\mathbb{F}\rvert}\right\} + \varepsilon_{\mathsf{gen}}$
Rounds	3
Alphabet	$\mathbb{F}$
Number of polynomials	$\lvert L_X\rvert + 3$
Oracle input queries	1
Proof queries	5
Randomness	$3\log\lvert\mathbb{F}\rvert + \mathsf{w}_{\mathsf{gen}}$
Verifier running time	$O(\log\lvert H\rvert + \widehat{\mathsf{t}}_{\mathsf{gen}})$
Max message degree	$\max\{d_X, d_Y, \lvert H\rvert - 2\}$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(1 - \sqrt{1-\delta})$ and $\widehat{\mathsf{t}}_{\mathsf{gen}}$ is defined as follows: For a seed s and $(\xi_j)_{j \in H} := \mathsf{Gen}(s)$, consider the polynomial $\hat{w} \in \mathbb{F}^{\leq |H|-1}[X]$ where $\hat{w}(j) = \xi_j$ for every $j \in H$. Then $\widehat{\mathsf{t}}_{\mathsf{gen}}$ is maximum time required to compute $\hat{w}(a)$ over every choice of s and $a \in \mathbb{F}$.

Construction 6.7. Let $f \in \mathcal{C} := \mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$ be a function and $\hat{f}$ be its extension to a polynomial with individual degrees d_X and d_Y. The honest prover $\mathbf{P}$ receives as input f, whereas the verifier $\mathbf{V}$ is given oracle access to f.

1. $\mathbf{P}$: Compute $\hat{f}$ and then, for every $i \in L_X$, send $\hat{r}_i(X) := \hat{f}(i, X) \in \mathbb{F}^{\leq d_Y}[X]$.
2. $\mathbf{V}$: Sample and send $s \leftarrow \{0,1\}^{\mathsf{w}_{\mathsf{gen}}}$.
3. $\mathbf{P}$: Let $(\xi_\ell)_{\ell \in H} \leftarrow \mathsf{Gen}(s)$ (formally, we associate each $x \in H$ with a unique index in $[|H|]$). Interpolate the points of $\vec{\xi}$ to define the weight polynomial $\hat{w} \in \mathbb{F}^{\leq |H|-1}[X]$ such that $\hat{w}(j) = \xi_j$ for every $j \in H$.Send $\hat{v} \in \mathbb{F}^{\leq d_X}[X]$ defined such that $\hat{v}(X) := \sum_{j \in H} \hat{w}(j) \cdot \hat{f}(X, j)$.
4. $\mathbf{V}$: Sample $i \leftarrow L_X$ uniformly at random and send to $\mathbf{P}$.
5. $\mathbf{P}$ and $\mathbf{V}$ run the interaction phase of the weighted univariate sumcheck protocol $(\mathbf{P}_\Sigma, \mathbf{V}_\Sigma)$ (as in Lemma 6.2) to show that $\sum_{j \in H} \hat{w}(j) \cdot \hat{r}_i(j) = \hat{v}(i)$:

$$\langle \mathbf{P}_\Sigma(d_Y, H, \hat{w}, v(i), \hat{r}_i), \mathbf{V}_\Sigma^{\hat{r}_i}(d_Y, H, \hat{w}, v(i)) \rangle \ ,$$

(Notice that this does not yet require $\mathbf{V}_\Sigma$ to know $\hat{v}(i)$ or compute or evaluate $\hat{w}$.)

6. **V**: Accept if and only if the following hold:
   - Run the decision phase of the univariate sumcheck (notice that this requires querying $v(i)$, and another 3 internal queries, and evaluating $\hat{w}$ on a random field element). Check that $\mathbf{V}_\Sigma$ accepts.
   - Sample $j \leftarrow L_Y^{(i)} := \{j \in \mathbb{F} \mid (i,j) \in D\}$ uniformly at random, query $\hat{r}_i(j)$ and $f(i,j)$, and check that $f(i,j) = \hat{r}_i(j)$.

Complexity parameters. We analyze the complexity parameters of the poly-IOPP.

- *Message degrees.* The rows $\hat{r}_i$ each have degree d_Y. The polynomial $\hat{v}$ has degree at most d_X. During the univariate sumcheck protocol, the prover sends a polynomial of degree $|H| - 2$ and a polynomial of degree d_Y.
- *Rounds.* The IOPP has 3 rounds.
- *Number of polynomials.* The prover begins by sending $|L_X|$ different polynomials $(\hat{r}_i)_{i \in L_X}$. It then sends a polynomial v and, in the univariate sumcheck, sends two additional polynomials. Thus the prover sends a total of $|L_X| + 3$ polynomials as oracles to the verifier.
- *Oracle input queries.* The verifier queries f in one location, $f(i,j)$.
- *Queries.* The verifier queries $r_i(j)$ and $v(i)$, and an additional three queries during the sumcheck protocol. The total proof query complexity is 5.
- *Randomness.* The verifier chooses s using $\mathsf{w}_{\mathsf{gen}}$ bits of randomness. It then queries i and j, requiring $\log|D| \leq 2\log|\mathbb{F}|$ bits of randomness. Finally, it uses $\log|\mathbb{F}|$ bits of randomness during the univariate sumcheck. The total randomness is therefore $3\log|\mathbb{F}| + \mathsf{w}_{\mathsf{gen}}$.
- *Verifier running time.* The verifier chooses s, and $O(1)$ field elements. It then runs the sumcheck verifier, and makes a constant number of comparisons between field elements. The running time of the sumcheck verifier is $O(\log|H|) + \widehat{\mathsf{t}}_{\mathsf{prx}}$. Observe that $\widehat{\mathsf{t}}_{\mathsf{prx}}$ dominates the time required to sample s since s is the input to Gen. The running time of the verifier is, therefore $O(\log|H| + \widehat{\mathsf{t}}_{\mathsf{prx}})$.

Proof of Lemma 6.6. We prove completeness and then proximity.

Completeness. Fix $f \in \mathcal{C}$. A transcript of the protocol has the following structure:

$$\mathsf{tr} = ((\hat{r}_i)_{i \in L_X}, s, \hat{v}, (i,j), \mathsf{tr}_\Sigma) \quad ,$$

where tr_Σ represents the transcripts of the sumcheck protocol executions. Let $(\xi_\ell)_{\ell \in H} := \mathsf{Gen}(s)$.

The honest prover generates $\hat{v}$ such that $\hat{v}(i) = \sum_{j \in H} \hat{w}(j) \cdot \hat{f}(i,j)$ for every $i \in L_X$, where $\hat{w}$ is the extension of the coefficients $(\xi_\ell)_{\ell \in H}$ to a degree $|H|-1$ polynomial. Observe that $\hat{w} \in \mathbb{F}^{\leq |H|-1}[X]$ and $\hat{f}(X,j) \in \mathbb{F}^{\leq d_X}[X]$ for every $j \in \mathbb{F}$. It follows that that $\hat{v} \in \mathbb{F}^{\leq d_X + |H| - 1}[X]$, and this will be sent by the honest prover. For every $i \in L_X$, it holds that $\sum_{j \in H} \hat{w}(j) \cdot \hat{r}_i(j) = \hat{v}(i)$. In other words, the claim for the univariate sumcheck protocol:

$$\langle \mathbf{P}_\Sigma(\mathcal{C}_\Sigma, H, \hat{w}, \hat{v}(i), \hat{r}_i), \mathbf{V}_\Sigma^{\hat{r}_i}(\mathcal{C}_\Sigma, H, \hat{w}, \hat{v}(i)) \rangle \quad ,$$

is true. Since the univariate sumcheck is perfectly complete, $\mathbf{V}_\Sigma$ will accept with probability 1.

Finally, it holds that $\hat{r}_i(j) = \hat{f}(i,j)$ for every $(i,j) \in D$. Thus no matter what point j is chosen by the verifier, its check will pass. We conclude that $\mathbf{V}$ accepts with probability 1.

Proximity. Fix f, a prover $\tilde{\mathbf{P}}$ and $\delta < 1 - \psi_{\mathsf{gen}}^2$. Set $\mu := \sqrt{1-\delta}$ and suppose that

$$\Pr\left[\langle \tilde{\mathbf{P}}, \mathbf{V}\rangle = 1\right] > \mu + \max\left\{\mu,\ \frac{d_Y + |H| - 1}{|\mathbb{F}|}\right\} + \varepsilon_{\mathsf{gen}}(1-\mu) \ .$$

We show that $\Delta(f, \mathcal{C}) \leq 1 - \mu^2 = \delta$.

A transcript of the protocol has the following structure:

$$\mathsf{tr} = ((\hat{r}_i)_{i \in L_X}, s, \hat{v}, (i,j), \mathsf{tr}_\Sigma) \ ,$$

where $(\hat{r}_i)_{i \in L_X}$ are polynomials of degree at most d_Y, $\hat{v}$ is of degree at most d_X, and tr_Σ is the transcript of the sumcheck protocol. Let $(\xi_\ell)_{\ell \in H} := \mathsf{Gen}(s)$ and let $c_1, \ldots, c_{|H|} \colon L_X \to \mathbb{F}$ be functions such that $c_j(i) = \hat{r}_i(j)$ for every $j \in H$.

Define sets S and $T \subseteq S$ as follows:

$$S := \left\{ i \in L_X \,\middle|\, |\{j \in L_Y^{(i)} \ :\ \hat{r}_i(j) = f(i,j)\}| \geq \mu \cdot |L_Y^{(i)}| \right\} \ ,$$

and

$$T := \left\{ i \in S \,\middle|\, \textstyle\sum_{j \in H} \xi_j \cdot \hat{r}_i(j) = \hat{v}(i) \right\} \ .$$

By definition, for every $i \in T$:

$$\hat{v}(i) = \sum_{j \in H} \xi_j \cdot \hat{r}_i(j) = \sum_{j \in H} \xi_j \cdot c_j(i) \ ,$$

We begin by showing that if T is small then the verifier is likely to reject.

Claim 6.8. $\Pr[\ \langle \mathbf{P}, \mathbf{V}\rangle = 1 \ \wedge\ |T| < \mu \cdot |L_X|\ ] \leq \mu + \max\left\{\mu, \frac{d_Y + |H| - 1}{|\mathbb{F}|}\right\}$.

Proof. The index i is chosen uniformly at random, and so $i \in T$ with probability less than μ, in which case we cannot bound the probability that it accepts from above. On the other hand, if $i \notin T$ then one of the following holds.

- $\hat{v}(i) \neq \sum_{j \in H} \xi_j \cdot \hat{r}_i(j)$: The prover and verifier run the sumcheck protocol to show that the sum of $\hat{r}_i$ over H is equal to $\hat{v}(i)$. Notice that $\hat{w} \in \mathbb{F}^{\leq |H|-1}[X]$ and $\hat{r}_i \in \mathbb{F}^{\leq d_Y}[X]$. Therefore by the soundness guarantee of the univariate sumcheck protocol, the verifier accepts with probability at most $(d_Y + |H| - 1)/|\mathbb{F}|$.
- $|\{j \in L_Y^{(i)} \ :\ \hat{r}_i(j) = f(i,j)\}| < \mu \cdot |L_Y^{(i)}|$: the verifier accepts only if it samples j with $\hat{r}_i(j) = f(i,j)$ which happens with probability at most μ.

Put together, we have that

$$\begin{aligned}
\Pr[\ \langle \mathbf{P}, \mathbf{V}\rangle = 1 \ \wedge\ |T| < \mu \cdot |L_X|\ ] &= \Pr[i \in T] \cdot \Pr[\ \langle \mathbf{P}, \mathbf{V}\rangle = 1 \ \wedge\ |T| < \mu \cdot |L_X| \mid i \in T\ ] \\
&\quad + \Pr[i \notin T] \cdot \Pr[\ \langle \mathbf{P}, \mathbf{V}\rangle = 1 \ \wedge\ |T| < \mu \cdot |L_X| \mid i \notin T\ ] \\
&\leq \mu + (1-\mu) \cdot \max\left\{\mu, \frac{d_Y + |H| - 1}{|\mathbb{F}|}\right\} \\
&< \mu + \max\left\{\mu, \frac{d_Y + |H| - 1}{|\mathbb{F}|}\right\} \ .
\end{aligned}$$

□

It follows from Claim 6.8 that,

$$\Pr[\,|T| \geq \mu \cdot |L_X|\,] \geq \Pr[\,\langle \mathbf{P}, \mathbf{V}\rangle = 1 \ \wedge\ |T| \geq \mu \cdot |L_X|\,] \geq \varepsilon_{\mathsf{gen}}(1-\mu)\ .$$

Observe that by the definition of $T \subseteq S$ there exists a $u \in \mathcal{C}_X$ (namely, $u = \hat{v}(L_X)$) where $u(T) = \sum_{j\in H} \xi_j \cdot c_j(T)$. Therefore,

$$\begin{aligned}
&\Pr\left[\begin{array}{l|r} \exists T \subseteq S & \\ |T| \geq \mu \cdot |L_X| & :\ \exists\, u \in \mathcal{C}_X,\ u(T) = \sum_{j\in H} \xi_j \cdot c_j(T) \end{array}\ \middle|\ \begin{array}{r} s \leftarrow \{0,1\}^{\mathsf{w}_{\mathsf{gen}}} \\ (\xi_\ell)_{\ell\in H} := \mathsf{Gen}(s) \end{array}\right] \\
&> \Pr[\,|T| \geq \mu \cdot |L_X|\,] \\
&\geq \varepsilon_{\mathsf{gen}}(1-\mu)\ .
\end{aligned}$$

Since $0 < \delta < 1 - \psi_{\mathsf{gen}}^2$ and $\delta := 1 - \mu^2$, it follows that $0 < 1 - \mu < 1 - \psi_{\mathsf{gen}}$. Since Gen is a strong proximity generator with proximity bound ψ_{gen} and error $\varepsilon_{\mathsf{gen}}$, we conclude that there exists a set $W \subseteq S$ with $|W| \geq \mu \cdot |L_X|$ such that

$$\forall\, j \in H,\ \exists\, u \in \mathcal{C}_X,\ u(W) = c_j(W)\ .$$

We now show that there exists a bivariate polynomial Q that completely agrees with rows $\hat{r}_i$ for every $i \in W$.

Claim 6.9. *There exists a bivariate polynomial $\hat{Q} \in \mathbb{F}[X,Y]$ with $\mathsf{deg}_X(\hat{Q}) \leq d_X$ and $\mathsf{deg}_Y(\hat{Q}) \leq d_Y$ such that $\hat{Q}(i,j) = \hat{r}_i(j)$ for every (i,j) where $i \in W$ and $j \in L_Y^{(i)}$.*

Proof. Let $\hat{c}_1, \ldots, \hat{c}_m$ be polynomials where $\hat{c}_i$ agrees with c_i on W. There exist such polynomials because

$$\forall\, j \in H,\ \exists\, u \in \mathcal{C}_X,\ u(W) = c_j(W)\ .$$

Notice that $\hat{r}_i(j) = \hat{c}_j(i)$ for every $(i,j) \in W \times H$.

Let W' be the first d_X rows in W (if there are less than d_X rows then take all of W) and for every $i \in W'$ let $I_{i,W'} \in \mathbb{F}^{\leq d_X}[Y]$ be the indicator polynomial where $I_{i,W'}(i) = 1$ and $I_{i,W'}(j) = 0$ for every $j \in W' \setminus \{i\}$. Define the polynomial $\hat{Q} \in \mathbb{F}[X,Y]$:

$$\hat{Q}(X,Y) := \sum_{i\in W'} I_{i,W'}(X) \cdot \hat{r}_i(Y)\ .$$

Notice that $\mathsf{deg}_X(\hat{Q}) \leq d_X$, that $\mathsf{deg}_Y(\hat{Q}) \leq d_Y$, and that $\hat{Q}(i,j) = r_i(j) = \hat{c}_j(i)$ for every $(i,j) \in W' \times H$. Since $\mathsf{deg}_X(\hat{Q}), \mathsf{deg}(\hat{c}_j) \leq d_X$ and $\hat{Q}(i,j) = \hat{c}_j(i)$ for at least d_X points, it follows that $\hat{Q}(X,j) \equiv \hat{c}_j$. Due to the fact that $\hat{r}_i(j) = \hat{c}_j(i) = \hat{Q}(i,j)$ for every $i \in W \setminus W'$ and $j \in [m]$, it holds that $\hat{Q}(i,j) = \hat{r}_i(j)$ for every $(i,j) \in W \setminus W' \times H$.

Finally, $\hat{Q}(i,Y) \equiv \hat{r}_i(Y)$ for every $i \in W$, since $\hat{Q}(i,Y)$ and $\hat{r}_i(Y)$ both have degree d_Y and agree on $|H| \geq d_Y$ points. Therefore, $\hat{Q}(i,j) = \hat{r}_i(j)$ for every (i,j) where $i \in W$ and $j \in L_Y^{(i)}$. □

Finally, by applying Claim 6.9 and recalling that $i \in W \subseteq S$ only if at least a μ-fraction of the values $j \in L_Y^{(i)}$ has $f(i,j) = \hat{r}_i(j)$, we have that $f(i,j) = \hat{Q}(i,j)$ for at least $\mu^2 \cdot |D|$ points. Therefore, $\Delta(f,\mathcal{C}) \leq \Delta(f,\hat{Q}) \leq 1 - \mu^2$. □

6.3 poly-IOPP for RS codes

We construct a poly-IOPP for Reed–Solomon codes. In more detail, we show that Lemma 6.6 suffices to construct a proximity test for univariate polynomials where the prover's messages are of degree significantly lower than that of the original function. The following claim says that univariate polynomials can be represented by bivariate polynomials where the degree of each variable is smaller than that of the original polynomial. This will allow us to map the Reed–Solomon evaluation of a polynomial to a bivariate Reed–Muller code to which we can test proximity using Lemma 6.6.

Claim 6.10 ([BS06])**.** *Given a polynomial $\hat{q} \in \mathbb{F}[X]$:*

- *For every $\hat{f} \in \mathbb{F}[X]$ there exists a unique bivariate polynomial $\hat{Q} \in \mathbb{F}[X,Y]$ with $\mathsf{deg}_X(\hat{Q}) = \left\lfloor \mathsf{deg}(\hat{f})/\mathsf{deg}(\hat{q}) \right\rfloor$ and $\mathsf{deg}_Y(\hat{Q}) \leq \mathsf{deg}(\hat{q}) - 1$ such that $\hat{f}(Z) = \hat{Q}(\hat{q}(Z), Z)$. Moreover, $\hat{Q}$ can be computed efficiently given $\hat{f}$ and $\hat{q}$. Observe that if $\mathsf{deg}(\hat{f}) \leq t \cdot \mathsf{deg}(\hat{q}) - 1$ then $\mathsf{deg}_X(\hat{Q}) \leq t - 1$.*
- *For every $\hat{Q} \in \mathbb{F}[X,Y]$ with $\mathsf{deg}_X(\hat{Q}) \leq t - 1$ and $\mathsf{deg}_Y(\hat{Q}) \leq \mathsf{deg}(\hat{q}) - 1$, the polynomial $\hat{f}(Z) := \hat{Q}(\hat{q}(Z), Z)$ has degree $\mathsf{deg}(\hat{f}) \leq t \cdot \mathsf{deg}(\hat{q}) - 1$.*

We can now state and prove the main lemma of this section:

Lemma 6.11. *Consider the following ingredients:*

- *$\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ is a Reed–Solomon code.*
- *$\hat{q} \in \mathbb{F}^{\leq d_q}[X]$ is a polynomial where $d := t \cdot d_q - 1$ for $t \in \mathbb{N}$ and $D := \{(\hat{q}(j), j) \mid j \in L\}$ is an admissible domain with row-index domain $L_X = \{\hat{q}(j) \mid j \in L\}$.*
- *$H \subseteq \mathbb{F}$ is a multiplicative subgroup of $\mathbb{F}^*$.*
- *Gen is a strong proximity generator for $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, t-1]$ for $|H|$ functions with seed length $\mathsf{w}_{\mathsf{gen}}$, proximity bound ψ_{gen}, and error $\varepsilon_{\mathsf{gen}}$.*

If $|\mathbb{F}| \geq |L_X|^2$ and $|H| \geq d_q - 1$ then Construction 6.12 is a poly-IOPP for $\mathcal{C}$ with the following parameters:

poly-IOPP to show $\delta < 1 - \psi_{\mathsf{gen}}^2$ proximity to $\mathcal{C}$	
Proximity error	$\sqrt{1-\delta} + \max\left\{\sqrt{1-\delta},\ \frac{d_q+\lvert H\rvert-2}{\lvert\mathbb{F}\rvert}\right\} + \varepsilon_{\mathsf{gen}}$
Rounds	3
Alphabet	$\mathbb{F}$
Number of polynomials	$\lvert L_X\rvert + 3$
Input queries	1
Proof queries	5
Randomness	$3\log\lvert\mathbb{F}\rvert + \mathsf{w}_{\mathsf{gen}}$
Verifier running time	$O(\log\lvert H\rvert + \widehat{\mathsf{t}}_{\mathsf{gen}})$
Max message degree	$\max\{d_q - 1, t - 1, \lvert H\rvert - 2\}$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(1 - \sqrt{1-\delta})$ and $\widehat{\mathsf{t}}_{\mathsf{gen}}$ is defined as in Lemma 6.6.

Construction 6.12. Let $f \in \mathcal{C}$ be a function. The honest prover $\mathbf{P}$ receives as input f, whereas the verifier $\mathbf{V}$ is given oracle access to f. Define $f' \colon D \to \mathbb{F}$ to be the bivariate function such that $f'(\hat{q}(j), j) = f(j)$, and set $d_X := \lfloor d_f/d_q \rfloor = t - 1$ and $d_Y := d_q - 1$.

1. The parties run the bivariate poly-IOPP $(\mathbf{P}_{\mathsf{PIOP}}, \mathbf{V}_{\mathsf{PIOP}})$ described in Lemma 6.6 to test that f' is close to the Reed–Muller code $\mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$. For any query $(\hat{q}(t), t) \in D$ made by $\mathbf{V}_{\mathsf{PIOP}}$ to f', $\mathbf{V}$ queries $f(t)$ and returns the answer to $\mathbf{V}_{\mathsf{PIOP}}$.

2. $\mathbf{V}$ accepts if and only if $\mathbf{V}_{\mathsf{PIOP}}$ accepts.

Proof. We prove completeness and proximity.

Completeness. If $f \in \mathcal{C}$ then, by Claim 6.10, there exists a polynomial $\hat{Q}$ with $\mathsf{deg}_X(\hat{Q}) \leq d_X$ and $\mathsf{deg}_Y(\hat{Q}) \leq d_Y$ whose evaluation on D is equal to f'. Therefore $f' \in \mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$. By the perfect completeness of the test in Lemma 6.6, it follows that $\mathbf{V}_{\mathsf{PIOP}}$ accepts with probability 1. Consequently, $\mathbf{V}$ always accepts.

Proximity. Fix a function f, a prover $\tilde{\mathbf{P}}$ and $\delta < 1 - \psi_{\mathsf{gen}}^2$. Suppose that $\tilde{\mathbf{P}}$ causes $\mathbf{V}$ to accept with probability greater than

$$\sqrt{1-\delta} + \max\left\{\sqrt{1-\delta},\ \frac{d_q + |H| - 2}{|\mathbb{F}|}\right\} + \varepsilon_{\mathsf{gen}} \ .$$

By the proximity of the poly-IOPP for $\mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$ it follows that there exists a bivariate polynomial $\hat{Q}$ with individual degrees d_X and d_Y such that $\Delta(\hat{Q}(D), f') \leq \delta$. Let $\hat{p} \in \mathbb{F}[X]$ be the polynomial such that $\hat{p}(X) = \hat{Q}(\hat{q}(X), X)$. Observe that by Claim 6.10, $\mathsf{deg}(\hat{p}) \leq t \cdot d_q - 1 = d_f$, and that $\hat{p}(i) = \hat{Q}(\hat{q}(i), i) = f'(\hat{q}(i), i) = f(i)$ for a $(1-\delta)$-fraction of the locations. Consequently: $\Delta(f, \mathcal{C}) \leq \Delta(f, \hat{p}(L)) \leq \delta$. □

We derive the following corollary for Reed–Solomon codes evaluated over multiplicative subgroups with order that is a power of two:

Corollary 6.13. *Consider the following ingredients:*

- $\mathbb{F}$ *is a field,* L *and* H *are multiplicative subgroups of* $\mathbb{F}^*$ *where* $|H|$ *divides* $|L|$.
- Gen *is a strong proximity generator for* $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, t-1]$ *for* $|H|$ *functions with seed length* $\mathsf{w}_{\mathsf{gen}}$*, proximity bound* ψ_{gen}*, and error* $\varepsilon_{\mathsf{gen}}$*, where* $L_X := \{a^{|H|} \mid a \in L\}$ *and* $t \in \mathbb{N}$.

If $|\mathbb{F}| \geq \left(\frac{|L|}{|H|}\right)^2$ *then there is a poly-IOPP for* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, t \cdot |H| - 1]$ *with the following parameters:*

poly-IOPP to show $\delta \leq 1 - \psi_{\mathsf{gen}}^2$ proximity to $\mathcal{C}$	
Proximity error	$\sqrt{1-\delta} + \max\left\{\sqrt{1-\delta},\ 2 \cdot |H|/|\mathbb{F}|\right\} + \varepsilon_{\mathsf{gen}}$
Rounds	3
Alphabet	$\mathbb{F}$
Number of polynomials	$|L|/|H| + 3$
Input queries	1
Proof queries	4
Randomness	$3 \log |\mathbb{F}| + \mathsf{w}_{\mathsf{gen}}$
Verifier running time	$O(\log |H| + \widehat{\mathsf{t}}_{\mathsf{gen}})$
Max message degree	$\max\{t-1, |H|-1\}$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(1 - \sqrt{1-\delta})$ *and* $\widehat{\mathsf{t}}_{\mathsf{gen}}$ *is defined as in Lemma 6.6.*

Proof. We use the poly-IOPP described in Lemma 6.11 using $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, t \cdot |L| - 1]$, $\hat{q}(X) := X^{|H|}$, the multiplicative subgroup H, and the proximity generator Gen.

Observe that, $\mathsf{deg}(\hat{q}) = |H|$ and, since $|H|$ divides $|L|$, $L_X := \{a^{|H|} \mid a \in L\}$ has $|L_X| = |L|/|H|$. Let $D := \{(\hat{q}(j), j) = (j^{|H|}, j) \mid j \in L\}$. Since $|H|$ divides the order of $|L|$, the function $\hat{q}(X) = X^{|H|}$ has exactly $|H|$ inverses for every $a \in L_X$. Therefore D is admissible.

Finally, observe that $|\mathbb{F}| > \left(\frac{|L|}{|H|}\right)^2 = |L_X|^2$ and $|H| = \mathsf{deg}(\hat{q})$, and so all of the requirements for Lemma 6.11 hold. The parameters follow. □

6.4 Recursive construction of IOPP for RS codes

We now prove Theorem 6.1 showing an IOP of proximity for the Reed–Solomon code that is compatible with the inverse-polynomial soundness error regime.

6.4.1 Preliminary claims

We begin by showing that if there is an IOPP for Reed–Solomon codes with some degree, then there is an IOPP for codes for (roughly) the degree squared.

Claim 6.14. *Consider the following ingredients:*

- $\mathbb{F}$ *is a field,* L *and* H *are multiplicative subgroups of* $\mathbb{F}^*$ *where* $|H|$ *divides* $|L|$.
- *An IOPP for* $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$.

If $|\mathbb{F}| \geq \left(\frac{|L|}{|H|}\right)^2$ *and* $d_{\mathsf{prx}} \geq |H| - 1$, *then there is an IOPP for* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, |H|^2 - 1]$ *with the following parameters:*

	IOPP for $\mathcal{C}_{\mathsf{prx}}$	$\to$ IOPP to show $\delta < 1 - \rho$ proximity to $\mathcal{C}$
Proximity error	β_{prx}	$\max\left\{2\sqrt{1-\delta}, \beta_{\mathsf{prx}}\right\} + O\left(\frac{\lvert H\rvert^3 \cdot (1/\rho)^4 + d_{\mathsf{prx}}^2/\rho_{\mathsf{prx}}^4 + \lvert H\rvert/\rho \cdot d_{\mathsf{prx}} \cdot (1/\rho_{\mathsf{prx}})^2}{\lvert\mathbb{F}\rvert}\right)$
Rounds	$\mathsf{k}_{\mathsf{prx}}$	$\mathsf{k}_{\mathsf{prx}} + 7$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert L_{\mathsf{prx}}\rvert \cdot \lvert H\rvert/\rho) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{prx},f}$	1
Proof queries	$\mathsf{q}_{\mathsf{prx},\pi}$	$O(\mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$
Randomness	$\mathsf{r}_{\mathsf{prx}}$	$15 \log \lvert\mathbb{F}\rvert + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\mathsf{vt}_{\mathsf{prx}}$	$\tilde{O}(\lvert H\rvert + \mathsf{q}_{\mathsf{prx},f}) + \mathsf{vt}_{\mathsf{prx}}$

Above, $\rho := |H|^2/|L|$, $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}} + 1)/|L_{\mathsf{prx}}|$, *and* $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}})$.

Proof. We first instantiate a poly-IOPP for $\mathcal{C}$ and then compile it into an IOPP. Set $L_X := \{a^{|H|} \mid a \in L\}$ and $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, |H| - 1]$. Observe that $|L_X| = |L|/|H|$ and that the rate of $\mathcal{C}_X$ is $\rho_X := |H|/|L_X| = |H|^2/|L| = \rho$ where ρ is the rate of $\mathcal{C}$.

Consider the poly-IOPP for $\mathcal{C}$ described in Corollary 6.13 with $t := |H|$ using the following ingredients:

- The field $\mathbb{F}$ and multiplicative subgroups L and H (observe that $|H|$ divides $|L|$).

- Gen is the strong proximity generator for $\mathcal{C}_X$, described in Item 1 of Corollary 4.5 for $|H|$ functions, with proximity bound $\sqrt{\rho_X} = \sqrt{\rho}$, seed length $\log|\mathbb{F}|$, and error $O\left(\frac{|H|^3\cdot(1/\rho_X)^4}{|\mathbb{F}|}\right) = O\left(\frac{|H|^3\cdot(1/\rho)^4}{|\mathbb{F}|}\right)$. The time to compute the interpolation of the coefficients generated by Gen is $\widehat{\mathsf{t}}_{\mathsf{gen}} = \tilde{O}(|H|)$.[10]

Observe that $|\mathbb{F}| \geq \left(\frac{|L|}{|H|}\right)^2$ and so, as a result, the requirements for Corollary 6.13 have been met. The resulting poly-IOPP for $\mathcal{C}$ has the following parameters:

poly-IOPP to show $\delta \leq 1-\rho$ proximity to $\mathcal{C}$	
Proximity error	$2\sqrt{1-\delta} + O\left(\frac{|H|^3\cdot(1/\rho)^4}{|\mathbb{F}|}\right)$
Rounds	3
Alphabet	$\mathbb{F}$
Number of polynomials	$\lvert H\rvert/\rho + 3$
Input queries	1
Proof queries	4
Randomness	$4\log\lvert\mathbb{F}\rvert$
Verifier running time	$\tilde{O}(\lvert H\rvert)$
Max message degree	$\lvert H\rvert - 1$

Since the proof query complexity of the poly-IOPP is 4, the verifier makes queries to at most 4 of the messages sent by the prover.

We now compile the poly-IOPP for $\mathcal{C}$ into an IOPP for $\mathcal{C}$ using Theorem 5.1 (following Remark 5.2 with $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} := 4$) with $\gamma := 1 - 2\sqrt{\rho_{\mathsf{prx}}}$ and the following ingredients:

- The poly-IOPP for $\mathcal{C}$ described above;
- $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ is the IOPP for $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$ described in the claim statement;
- Gen is a proximity generator for $\mathcal{C}_{\mathsf{prx}}$. We use the generator described in Item 2 of Theorem 4.2 for 8 functions, seed length $8\log|\mathbb{F}|$, proximity bound $\psi_{\mathsf{gen}} := \sqrt{\rho_{\mathsf{prx}}}$, error $O\left(\frac{d_{\mathsf{prx}}^2/\rho_{\mathsf{prx}}^4}{|\mathbb{F}|}\right)$, and computation time $O(1)$.

Observe that $d_{\mathsf{prx}} \geq |H| - 1$, and so bounds the degrees of the prover messages in the poly-IOPP. It holds that $0 < \gamma < 1 - \max\{\psi_{\mathsf{gen}}, 2\rho_{\mathsf{prx}}\}$, and, by the Johnson bound Theorem 3.5, the code $\mathcal{C}_{\mathsf{prx}}$ is $(\gamma, 1/2\rho_{\mathsf{prx}})$–list decodable. The requirements for Theorem 5.1 have therefore been met. The resulting IOPP for $\mathcal{C}$ has proximity error

$$\max\left\{2\sqrt{1-\delta}, \beta_{\mathsf{prx}}\right\} + O\left(\frac{|H|^3\cdot(1/\rho)^4 + d_{\mathsf{prx}}^2/\rho_{\mathsf{prx}}^4 + |H|/\rho\cdot d_{\mathsf{prx}}\cdot(1/\rho_{\mathsf{prx}})^2}{|\mathbb{F}|}\right) \quad ,$$

where $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}})$. Its remaining parameters are:

	poly-IOPP for R	*IOPP for $\mathcal{C}_{\mathsf{prx}}$*	*$\rightarrow$ IOPP for R*
Rounds	3	$\mathsf{k}_{\mathsf{prx}}$	$\mathsf{k}_{\mathsf{prx}} + 7$
Alphabet	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\lvert H\rvert/\rho + 3$ *polynomials*	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert L_{\mathsf{prx}}\rvert\cdot\lvert H\rvert/\rho) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	1	$\mathsf{q}_{\mathsf{prx},f}$	1
Proof queries	4	$\mathsf{q}_{\mathsf{prx},\pi}$	$O(\mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$
Randomness	$4\log\lvert\mathbb{F}\rvert$	$\mathsf{r}_{\mathsf{prx}}$	$15\log\lvert\mathbb{F}\rvert + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\tilde{O}(\lvert H\rvert)$	$\mathsf{vt}_{\mathsf{prx}}$	$\tilde{O}(\lvert H\rvert + \mathsf{q}_{\mathsf{prx},f}) + \mathsf{vt}_{\mathsf{prx}}$

[10] The time to compute the coefficients themselves is $O(|H|)$. Following the computation of the $|H|$ coefficients, the time to evaluate the polynomial interpolating these points on a random field element is at most $\tilde{O}(|H|)$.

□

We now use Claim 6.14 to recursively construct IOPPs for Reed–Solomon codes for any degree of a specific structure.

Lemma 6.15. *Consider the following ingredients:*

- $m \in \mathbb{N}$ *is a degree parameter.*
- $\mathbb{F}$ *is a field and suppose that* $\mathbb{F}^*$ *contains a multiplicative subgroup* L *of order* 2^k*.*

If $|\mathbb{F}| \geq 2^{2k}$ *and* $k \geq 2m$ *then, letting* $d := 2^{2m} - 1$*, then Construction 6.16 is an IOPP for* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ *with the following parameters:*

IOPP to show δ proximity to $\mathcal{C}$	
Proximity error	$\max\left\{2\sqrt{1-\delta},\ \rho^{1/4}\right\} + O\left(\frac{d^{3/2}\cdot(1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)$
Rounds	$O(\log\log d)$
Alphabet	$\mathbb{F}$
Proof length	$O(\lvert L\rvert/\rho)$
Oracle input queries	1
Proof queries	$O(\log\log d)$
Randomness	$O(\log\log d \cdot \log\lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(\sqrt{d})$

Construction 6.16. Let g be a generator of L. On input $f\colon L \to \mathbb{F}$, the protocol is as follows:

1. *If* $m < 7$*:* **P** and **V** run the low-degree test for constant degree to described in Claim 3.6 to show that f is close to $\mathcal{C}$.
2. *If* $m \geq 7$*:* Let $d_{\mathsf{prx}} := 2^m - 1$ if m is even, and $d_{\mathsf{prx}} := 2^{m+1} - 1$ otherwise, and let $L_{\mathsf{prx}} := \langle g^{2^{m-7}} \rangle$. **P** and **V** run the IOPP described in Claim 6.14 to show proximity of f to $\mathcal{C}$, with $H = \langle g^{2^{k-m}} \rangle$ and using an IOPP for $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$ that is instantiated recursively.

Proof. We prove the lemma by induction on the degree exponent parameter m.

Basis. When $m < 7$, by Claim 3.6 we have an IOPP with the following parameters:

IOPP to show δ proximity to $\mathcal{C}$ with constant degree	
Proximity error	$1 - \delta$
Rounds	1
Alphabet	$\mathbb{F}$
Proof length	$O(1)$
Input queries	1
Proof queries	$O(1)$
Randomness	$\log\lvert\mathbb{F}\rvert$
Verifier running time	$O(1)$

Induction step. Suppose that for every $m' < m$ the lemma holds. We show that this holds for m. Let $d := 2^{2m} - 1$ and L be a multiplicative subset of $\mathbb{F}^*$ of order 2^k with $|\mathbb{F}| \geq 2^{2k}$. Define d_{prx}, L_{prx} and H as in Item 2 of Construction 6.16 with respect to m and L. Observe the following:

- $|H| = 2^m$ divides $|L| = 2^k$;
- $d = |H|^2 - 1$;
- $d_{\mathsf{prx}} := 2^{2m'} - 1$ for $m' \in \mathbb{N}$;

- $|L_{\mathsf{prx}}| = 2^{k-m+7} = O(|L|/|H|)$;
- $O(\log\log d_{\mathsf{prx}} + 1) = O(\log\log d)$;
- $2^{-7} \cdot \rho \leq \rho_{\mathsf{prx}} \leq 2^{-6} \cdot \rho$.

By plugging in m' and L_{prx} into the induction assumption, the code $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$ has an IOPP with the following parameters:

IOPP to show δ proximity to $\mathcal{C}_{\mathsf{prx}}$	
Proximity error	$\max\left\{2\sqrt{1-\delta}, \rho_{\mathsf{prx}}^{1/4}\right\} + O\left(\frac{d_{\mathsf{prx}}^{3/2} \cdot (1/\rho_{\mathsf{prx}})^4}{\lvert\mathbb{F}\rvert}\right)$
Rounds	$O(\log\log d_{\mathsf{prx}})$
Alphabet	$\mathbb{F}$
Proof length	$O(\lvert L_{\mathsf{prx}}\rvert/\rho_{\mathsf{prx}})$
Oracle input queries	1
Proof queries	$O(\log\log d_{\mathsf{prx}})$
Randomness	$O(\log\log d_{\mathsf{prx}} \cdot \log\lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(\sqrt{d_{\mathsf{prx}}})$

Now apply Claim 6.14 using the IOPP for $\mathcal{C}_{\mathsf{prx}}$ as in Item 2. The result of this is an IOPP for $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ with $d := 2^{2m} - 1$ with the following parameters:

- ***Proximity error.*** The proximity error of the IOPP for $\mathcal{C}_{\mathsf{prx}}$, when $\delta = 1 - 2\sqrt{\rho_{\mathsf{prx}}}$ is

$$\begin{aligned}
\beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}}) &= \max\left\{2 \cdot \left(2 \cdot \rho_{\mathsf{prx}}^{1/2}\right)^{1/2}, \rho_{\mathsf{prx}}^{1/4}\right\} + O\left(\frac{d_{\mathsf{prx}}^{3/2} \cdot (1/\rho_{\mathsf{prx}})^4}{|\mathbb{F}|}\right) \\
&\leq \max\left\{2 \cdot \left(2 \cdot \left(2^{-6} \cdot \rho\right)^{1/2}\right)^{1/2}, \left(2^{-6} \cdot \rho\right)^{1/4}\right\} + O\left(\frac{d^{3/4} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \\
&= \rho^{1/4} + O\left(\frac{d^{3/4} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \quad .
\end{aligned}$$

  The proximity error of the resulting IOPP is

$$\begin{aligned}
&\max\left\{2\sqrt{1-\delta},\ \beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}})\right\} + O\left(\frac{|H|^3 \cdot (1/\rho)^4 + d_{\mathsf{prx}}^2/\rho_{\mathsf{prx}}^4 + |H|/\rho \cdot d_{\mathsf{prx}} \cdot (1/\rho_{\mathsf{prx}})^2}{|\mathbb{F}|}\right) \\
&= \max\left\{2\sqrt{1-\delta},\ \beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}})\right\} + O\left(\frac{d^{3/2} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \quad .
\end{aligned}$$

  We can now plug in the value of $\beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}})$ to get error at most:

$$\begin{aligned}
&\max\left\{2\sqrt{1-\delta},\ \rho^{1/4} + O\left(\frac{d^{3/4} \cdot (1/\rho)^4}{|\mathbb{F}|}\right)\right\} + O\left(\frac{d^{3/2} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \\
&= \max\left\{2\sqrt{1-\delta},\ \rho^{1/4}\right\} + O\left(\frac{d^{3/2} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \quad .
\end{aligned}$$

  Observe that, while the poly-IOPP for $\mathcal{C}$ has the soundness error given only when $\delta < 1 - \rho$, the above soundness error is greater than ρ, and so we can remove this restriction.

- ***Rounds.*** The number of rounds is $O(\log\log d_{\mathsf{prx}}) + 7 = O(\log\log d)$.

- ***Proof length.*** The proof length is

$$O\left(|L_{\mathsf{prx}}| \cdot \frac{|H|}{\rho} + \frac{|L_{\mathsf{prx}}|}{\rho_{\mathsf{prx}}}|\right) = O\left(\frac{|L|}{|H|} \cdot \frac{|H|}{\rho} + \frac{|L|}{|H| \cdot \rho}\right) = O\left(\frac{|L|}{\rho}\right) \quad .$$

- ***Oracle input queries.*** The verifier makes 1 query to its input.
- ***Proof queries.*** The proof query complexity is $O(\log\log d_{\mathsf{prx}}) + O(1) = O(\log\log d)$.
- ***Randomness.*** The randomness complexity is $O(\log\log d_{\mathsf{prx}} \cdot \log|\mathbb{F}|) + 15\log|\mathbb{F}| = O(\log\log d \cdot \log|\mathbb{F}|)$.
- ***Verifier running time.*** The verifier running time is $\tilde{O}(|H|) + \tilde{O}(\sqrt{d_{\mathsf{prx}}}) = \tilde{O}(\sqrt{d})$.

□

6.4.2 Proof of Theorem 6.1

Let m be the smallest integer such that $d \leq 2^{2m} - 1$ and $L_{\mathsf{prx}} := G$, and set $d_{\mathsf{prx}} := 2^{2m} - 1$. Observe that $d \leq d_{\mathsf{prx}} < 4d$, that $|L_{\mathsf{prx}}| \geq 2^8 \cdot |L|$. We apply Theorem 8.2 with the following ingredients:

- The Reed–Solomon code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$.
- The IOPP for $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$ described in Lemma 6.15. Notice that the rate of $\mathcal{C}_{\mathsf{prx}}$ is $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}} + 1)/|L_{\mathsf{prx}}| \leq 2^{-6} \cdot \rho$.
- The proximity generator for $\mathcal{C}_{\mathsf{prx}}$ described in Item 2 of Theorem 4.2 for 2 functions with seed length $2\log|\mathbb{F}|$, proximity bound $\sqrt{\rho_{\mathsf{prx}}}$, error $O\left(\frac{d_{\mathsf{prx}}^2 \cdot (1/\rho_{\mathsf{prx}})^4}{|\mathbb{F}|}\right)$ and computation time $O(1)$.
- $\gamma := 1 - 2\sqrt{\rho_{\mathsf{prx}}}$.

Observe that $d \leq d_{\mathsf{prx}}$, that $0 < \gamma < 1 - \max\{\sqrt{\rho_{\mathsf{prx}}}, 2\rho_{\mathsf{prx}}\}$ and that, by the Johnson bound Theorem 3.5, $\mathcal{C}_{\mathsf{prx}}$ is $(\gamma, 1/2\rho_{\mathsf{prx}})$–list decodable. We can therefore apply Theorem 8.2 to get an IOPP for $\mathcal{C}$ with the following parameters:

- ***Proximity error.*** The proximity error of the IOPP for $\mathcal{C}_{\mathsf{prx}}$, when $\delta = 1 - 2\sqrt{\rho_{\mathsf{prx}}}$ is

$$\begin{aligned}
\beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}}) &= \max\left\{2\sqrt{1 - (1 - 2\sqrt{\rho_{\mathsf{prx}}})},\ \rho_{\mathsf{prx}}^{1/4}\right\} + O\left(\frac{d_{\mathsf{prx}}^{3/2} \cdot (1/\rho_{\mathsf{prx}})^4}{|\mathbb{F}|}\right) \\
&\leq \max\left\{2 \cdot \left(2 \cdot \left(2^{-6} \cdot \rho\right)^{1/2}\right)^{1/2}, \left(2^{-6} \cdot \rho\right)^{1/4}\right\} + O\left(\frac{d^{3/2} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \\
&= \rho^{1/4} + O\left(\frac{d^{3/2} \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \quad .
\end{aligned}$$

  The proximity error of the resulting IOPP is therefore at most

$$\max\left\{1 - \delta, \beta_{\mathsf{prx}}(1 - 2\sqrt{\rho_{\mathsf{prx}}}) + O\left(\frac{d_{\mathsf{prx}}^2 \cdot (1/\rho_{\mathsf{prx}})^2}{|\mathbb{F}|}\right)\right\} = \max\left\{1 - \delta, \rho^{1/4} + O\left(\frac{d^2 \cdot (1/\rho)^4}{|\mathbb{F}|}\right)\right\} \quad .$$

- ***Rounds.*** The number of rounds is $O(\log\log d_{\mathsf{prx}}) + 1 = O(\log\log d)$.

- ***Proof length.*** The proof length is $O\left(|L_{\mathsf{prx}}| + |L_{\mathsf{prx}}|/\rho_{\mathsf{prx}}\right) = O(|L|/\rho)$.
- ***Oracle input queries.*** The verifier makes 1 query to its input.
- ***Proof queries.*** The proof query complexity is $O(\log\log d_{\mathsf{prx}}) + 1 = O(\log\log d)$.
- ***Randomness.*** The randomness complexity is $O(\log\log d_{\mathsf{prx}} \cdot \log|\mathbb{F}|) + O(\log|\mathbb{F}|) = O(\log\log d \cdot \log|\mathbb{F}|)$.
- ***Verifier running time.*** The verifier running time is $\tilde{O}(\sqrt{d_{\mathsf{prx}}}) + O(1) = \tilde{O}(\sqrt{d})$.

7 High-soundness IOP for NP

We show that NP has an IOP with small soundness error and small query complexity. We first state our result, which will depend on a poly-IOP for NP, which we describe in Section 7.1.

The NP-complete language of choice for our IOP is the R1CS relation:

Definition 7.1. *The* R1CS *relation* R_{R1CS} *is the set of all pairs* $((\mathbb{F}, k, n, m, A, B, C, u), w)$ *where* $\mathbb{F}$ *is a finite field,* $k, n, m \in \mathbb{N}$ *denote the number of inputs, variables and number of non-zero entries respectively,* A,B *and* C *are* $n \times n$ *matrices over* $\mathbb{F}$, $u \in \mathbb{F}^k$ *and* $w \in \mathbb{F}^{n-k}$ *such that for all* $i \in [n]$

$$\left(\sum_{j=1}^{n} A_{i,j} \cdot z_j\right) \cdot \left(\sum_{j=1}^{n} B_{i,j} \cdot z_j\right) = \sum_{j=1}^{n} C_{i,j} \cdot z_j \ ,$$

where $z := (u, w) \in \mathbb{F}^n$. *Here* m *is the maximum number of non-zero entries in* A, B *and* C.

Our main theorem is an IOP for R1CS with inverse-polynomial soundness error and loglog query complexity:

Theorem 7.2. *Let* $n, t \in \mathbb{N}$ *and* $\beta \in (0, 1)$ *be parameters and* $\mathbb{F}$ *be a field. If* $\mathbb{F}^*$ *contains a multiplicative subgroup* G *whose order is a power of two and* $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot n \cdot \beta^{-17/8t}$ *then there is an IOP for* R_{R1CS} *over the field* $\mathbb{F}$ *with* n *inputs with the following parameters:*

IOP for R1CS	
Soundness error	β
Rounds	$O(\log\log n)$
Alphabet	$\mathbb{F}$
Proof length	$O\left(t \cdot n \cdot \beta^{-1/t}\right)$
Queries	$O(t \cdot \log\log n)$
Randomness	$O(t \cdot \log\log n \cdot \log \lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(t \cdot (m + n))$

Proof. We combine the poly-IOP for R1CS described in Lemma 7.3, with Theorem 8.1, setting $\rho := \beta^{1/2t}/16$. This yields an IOP for R1CS with the following parameters:

- ***Proximity error.*** The proximity error of the IOP is

$$\begin{aligned}
\beta &:= \max\left\{O\left(\frac{n}{|\mathbb{F}|}\right),\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{n^2 \cdot (1/\rho)^4}{|\mathbb{F}|}\right)\right\} \\
&= \rho^{1/4} + O\left(\frac{n^2 \cdot (1/\rho)^4}{|\mathbb{F}|}\right) \\
&= O\left(\beta^{1/8t} + \frac{n^2 \cdot \beta^{-2/t}}{|\mathbb{F}|}\right) \\
&= O(\beta^{1/8t}) \ ,
\end{aligned}$$

  where the first equality follows since $\rho < 1/16$, and so $2\rho^{1/2} < \rho^{1/4}$ and the final equality follows from the fact that $|\mathbb{F}| \geq \Omega(n^2 \cdot \beta^{2/t} \cdot \beta^{1/8t})$.

- ***Rounds.*** The number of rounds is $O(\log\log n)$.

- ***Proof length.*** The proof length is $O(n/\rho^2) = O(n/\beta^{1/t})$.
- ***Proof queries.*** The proof query complexity is $O(\text{loglog}\, n)$.
- ***Randomness.*** The randomness complexity is $O\left(\text{loglog}\, n \cdot \log |\mathbb{F}|\right)$.
- ***Verifier running time.*** The verifier running time is $O(n+m) + \tilde{O}(\sqrt{n}) = O(n+m)$.

Finally, we repeat the protocol $O(t)$ times in parallel to get the soundness error down from $O(\beta^{1/8t})$ to β. □

7.1 poly-IOP for R1CS

In this section we construct a poly-IOP for R1CS with inverse-polynomial soundness error. This is a simplified version of the poly-IOP for R1CS described in [BCRSVW19].

Lemma 7.3. *There exists a poly-IOP for the* NP*-complete relation* R1CS *with the following parameters:*

poly-IOP for R1CS	
Soundness error	$2 \cdot (n-1)/\lvert\mathbb{F}\rvert$
Rounds	2
Alphabet	$\mathbb{F}$
Number of polynomials	11
Queries	14
Randomness	$3 \log \lvert\mathbb{F}\rvert$
Verifier running time	$\tilde{O}(m+n)$
Max message degree	$n-1$

We describe the construction:

Construction 7.4. Given a field $\mathbb{F}$ let H be a subgroup of $\mathbb{F}^*$ of order n. We sometimes refer to elements of H as elements in $[|H|]$, formally, this is done by defining a bijection between the two and using it as appropriate to translate between the two domains. Let $H_{\mathsf{in}} \subseteq H$ be the subset of order $|H_{\mathsf{in}}| = k$ that corresponds to the indices $\{1, \ldots, k\}$. Finally, let $\hat{V}_H$ be the unique non-zero polynomial of degree at most $n-1$ that is 0 on H. Define $\hat{V}_{H_{\mathsf{in}}}$ similarly with regards to H_{in}.

On input an R1CS instance $((\mathbb{F}, k, n, m, A, B, C, v), w)$, where the prover is given the entire instance, and the verifier is given $(\mathbb{F}, k, n, m, A, B, C, v)$, the protocol proceeds as follows:

1. **P**: send the polynomials $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h}, \hat{f}_w \in \mathbb{F}^{\leq |H|-1}[X]$ defined as follows:
   (a) let $z := (u, w) \in \mathbb{F}^n$. For every $M \in \{A, B, C\}$, $\hat{f}_M$ is the unique polynomial with $\hat{f}_M(X) := Mz(X)$ for every $x \in H$.
   (b) $\hat{h}(X) := \frac{\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X)}{\hat{V}_H(X)}$.
   (c) $\hat{f}_w(X) := \frac{w(X) - \hat{u}(X)}{\hat{V}_{H_{\mathsf{in}}(X)}}$ where $\hat{u}$ is the unique degree $n-1$ polynomial that is equal to u on H_{in} and 0 on H_{aux}.
2. **V**: choose $r \leftarrow \mathbb{F}$ uniformly at random and send to **P**. Define the following:
   (a) $\hat{p}_r \in \mathbb{F}^{\leq |H|-1}[X]$ is the unique polynomial such that $\hat{p}_r(x) := r^x$ for every $x \in H$.

(b) for every $M \in \{A, B, C\}$, $\hat{q}_{M,r} \in \mathbb{F}^{\leq |H|-1}[X]$ is the unique polynomial such that $\hat{q}_{M,r}(X) := \sum_{a \in H} M^{\top}(\alpha, X) \cdot r^X$ for every $x \in H$.

3. **P** and **V** execute the poly-IOP for univariate sumcheck (Lemma 6.2) for every $M \in \{A, B, C\}$ (in parallel and with shared randomness) to show that

$$\sum_{\alpha \in H} \hat{p}_r(\alpha) \cdot \hat{f}_M(\alpha) - \hat{q}_{M,r}(\alpha) \cdot \hat{f}_z(\alpha) = 0 \ .$$

where if $\mathbf{V}_\Sigma$ makes a query α to $\hat{f}_z$, **V** returns $\hat{f}_z(\alpha) := \hat{f}_w(\alpha) \cdot \hat{V}_{H_{\mathsf{in}}}(\alpha) + \hat{u}(\alpha)$ by making a single query to $\hat{f}_w$, where $\hat{u}$ is defined as before.

4. **V**: sample $a \leftarrow \mathbb{F}$ and accepts if and only if the sumcheck verifier accepted in every execution and $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{h}(a) \cdot \hat{V}_H(a)$.

Proof of Lemma 7.3. We prove completeness, then soundness, and finally analyze complexity measures. Completeness and soundness rely on the fact that for every $r \in \mathbb{F}$ it holds that:

$$\begin{aligned}
\sum_{\alpha \in H} \hat{p}_r(\alpha) \cdot \hat{f}_M(\alpha) - \hat{q}_{M,r}(\alpha) \cdot \hat{f}(\alpha) &= \sum_{\alpha \in H} r^\alpha \cdot \hat{f}_M(\alpha) + \sum_{\alpha \in H} \sum_{\beta \in H} r^\alpha \cdot M^{\top}(\alpha, \beta) \cdot \hat{f}_z(\alpha) \\
&= \sum_{\alpha \in H} r^\alpha \cdot \hat{f}_M(\alpha) + \sum_{\alpha \in H} \sum_{\beta \in H} r^\alpha \cdot M^{\top}(\beta, \alpha) \cdot \hat{f}_z(\beta) \\
&= \sum_{\alpha \in H} \left(\hat{f}_M(\alpha) - \sum_{\beta \in H} M^{\top}(\beta, \alpha) \cdot \hat{f}_z(\beta) \right) \cdot r^\alpha \\
&= \sum_{\alpha \in H} \left(\hat{f}_M(\alpha) - \sum_{\beta \in H} M(\alpha, \beta) \cdot \hat{f}_z(\beta) \right) \cdot r^\alpha \qquad (3)
\end{aligned}$$

Completeness. Fix an instance $((\mathbb{F}, k, n, m, A, B, C, u), w) \in R_{\mathsf{R1CS}}$. Since $Az + Bz - Cz = 0$ it holds that the polynomial $\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X)$ is zero over H, and therefore the polynomial $\hat{V}_H$ divides it. Therefore the polynomial

$$\hat{h}(X) = \frac{\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X)}{\hat{V}_H(X)} \ ,$$

is well defined. It holds that $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{h}(a) \cdot \hat{V}_H(a)$ for every $a \in \mathbb{F}$. Consequently, the verifier will always accept during its check that $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{h}(a) \cdot \hat{V}_H(a)$.

We now show that the verifier will accept with probability 1 in the sumcheck executions. Consider $M \in \{A, B, C\}$ and fix $r \in \mathbb{F}$. The sumcheck protocol checks that

$$\sum_{\alpha \in H} \hat{p}_r(\alpha) \cdot \hat{f}_M(\alpha) - \hat{q}_{M,r}(\alpha) \cdot \hat{f}(\alpha) = 0 \ .$$

By Equation 3 and by plugging in the definition of $\hat{f}_z$ we can rewrite

$$\begin{aligned}\sum_{\alpha\in H}\hat{p}_r(\alpha)\cdot\hat{f}_M(\alpha)-\hat{q}_{M,r}(\alpha)\cdot\hat{f}_z(\alpha) &= \sum_{\alpha\in H}\left(\hat{f}_M(\alpha)-\sum_{\beta\in H}M(\alpha,\beta)\cdot\hat{f}_z(\beta)\right)\cdot r^\alpha \\ &= \sum_{\alpha\in H}\left(\hat{f}_M(\alpha)-\sum_{\beta\in H}M(\alpha,\beta)\cdot\left(\hat{f}_w(\beta)\cdot\hat{V}_{H_{\mathsf{in}}}(\beta)+\hat{u}(\beta)\right)\right)\cdot r^\alpha \\ &= \sum_{\alpha\in H}\left(\hat{f}_M(\alpha)-Mz(\alpha)\right)\cdot r^\alpha \\ &= 0\ .\end{aligned}$$

It follows that the sumcheck verifier will accept with probability 1.

Soundness. Fix an instance $(\mathbb{F},k,n,m,A,B,C,u)\notin L(R_{\mathsf{R1CS}})$ and a prover $\tilde{\mathbf{P}}$. Let $\hat{f}_A,\hat{f}_B,\hat{f}_C,\hat{h},\hat{f}_w\in\mathbb{F}^{\leq|H|-1}[X]$ be the polynomials sent by the prover, and set $\hat{f}_z(X):=\hat{f}_w(X)\cdot\hat{V}_{H_{\mathsf{in}}}(X)+\hat{u}(X)$.

If it holds that

$$\hat{f}_A(X)\cdot\hat{f}_B(X)-\hat{f}_C\neq\hat{h}(X)\cdot\hat{V}_H(X)\ ,$$

then, by the Lemma 3.8, $\mathbf{V}$ with probability at least $1-(|H|-1)/|\mathbb{F}|$ over the choice of $a\leftarrow L$:

$$\hat{f}_A(a)\cdot\hat{f}_B(a)-\hat{f}_C(a)\neq\hat{h}(a)\cdot\hat{V}_H(a)\ ,$$

causing $\mathbf{V}$ to reject.

Suppose, then, that

$$\hat{f}_A(X)\cdot\hat{f}_B(X)-\hat{f}_C(X)=\hat{h}(X)\cdot\hat{V}_H(X)\ ,$$

which means that $\hat{f}_A(x)\cdot\hat{f}_B(x)=\hat{f}_C(x)$ for any $x\in H$. Let $z(x):=\hat{f}_z(x)$ for every $x\in H$. Notice that $z=(u,w)\in\mathbb{F}^n$ for some w. Since $(\mathbb{F},k,n,A,B,C,u)\notin L(R_{\mathsf{R1CS}})$, it holds that for some $\alpha\in H$, $(Az)(\alpha)+(Bz)(\alpha)\neq(Cz)(\alpha)$. Since $\hat{f}_A(x)\cdot\hat{f}_B(x)=\hat{f}_C(x)$ for any $x\in H$, it follows that for some $M\in\{A,B,C\}$, $\hat{f}_M(X)$ is not the extension of Mz, i.e., there exists $\alpha\in H$ where $\hat{f}_M(\alpha)\neq(Mz)(\alpha)$.

Define $\hat{g}$ as follows:

$$\hat{g}(X):=\sum_{\alpha\in H}\left(\hat{f}_M(\alpha)-\sum_{\beta\in H}M(\alpha,\beta)\cdot\hat{f}(\beta)\right)\cdot X^\alpha\ .$$

Since $\hat{f}_M(\alpha)\neq Mz(\alpha)$ for some $\alpha\in H$, $\hat{g}$ is not the zero polynomial. Moreover, the degree of $\hat{g}$ is at most $|H|-1$. Thus, with probability at least $1-(|H|-1)/|L|$ over the choice of r, $\hat{g}(r)\neq 0$. If this is the case, by Equation 3 we can write

$$\begin{aligned}0\neq\hat{g}(r) &= \sum_{\alpha\in H}\left(\hat{f}_M(\alpha)-\sum_{\beta\in H}M(\alpha,\beta)\cdot\left(\hat{f}_w(\beta)\cdot\hat{V}_{H_{\mathsf{in}}}(\beta)+\hat{u}(\beta)\right)\right)\cdot r^\alpha \\ &= \sum_{\alpha\in H}\hat{p}_r(\alpha)\cdot\hat{f}_M(\alpha)-\hat{q}_{M,r}(\alpha)\cdot\hat{f}(\alpha)\ .\end{aligned}$$

It follows that the sumcheck claim is false, in which case the sumcheck verifier $\mathbf{V}_\Sigma$ will reject with probability at least $1-(|H|-1)/|L|$.

We conclude that $\mathbf{V}$ accepts with probability at most $2(|H|-1)/|L| = 2 \cdot (n-1)/|L|$.

Complexity parameters. We analyze the complexity parameters of the IOP.

- *Rounds.* The protocol has two rounds.
- *Number of polynomials.* The IOP prover sends 11 polynomials: $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h}, \hat{f}_w$ constitute 5, and each of the 3 sumcheck claim requires the prover to send 2 polynomials.
- *Queries.* $\mathbf{V}$ makes 14 queries to prover messages: it makes one query to each $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{f}_w$ during the the sumcheck executions, and one query to each of 6 internal messages in the sumcheck protocols (2 for each execution) and one query to each $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h}$ in its final test.
- *Randomness.* $\mathbf{V}$ uses $3 \log |\mathbb{F}|$ bits of randomness: it chooses $r \in \mathbb{F}$, uses $\log |\mathbb{F}|$ bits of randomness in the sumcheck protocols, and chooses $a \in \mathbb{F}$.
- *Verifier running time.* The verifier runs in time $\tilde{O}(m+n)$.
- *Max message degree.* Each of the polynomials $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h}, \hat{f}_w$ has degree at most $n-1$. The messages sent during the univariate sumcheck protocol have degrees $n-1$ and $n-2$. Therefore the maximal degree is $n-1$.

□

8 Applications

In this section we describe further applications of our theorems:

- In Section 8.1 we plug in our proximity test for Reed–Solomon codewords into the generic compiler described in Section 5, giving a compiler from poly-IOPPs to IOPPs that can be used with any poly-IOPP.
- In Section 8.2 we show that, using our techniques, a proximity test for Reed–Solomon codes that works on specific evaluation domains can be adapted to work for *any* evaluation domain.
- In Section 8.3 we give a high-soundness small-query proximity test for bivariate Reed–Muller codes.

8.1 poly-IOPPs to IOPPs

In this section we utilize our proximity test for Reed–Solomon codes to compile any poly-IOPP (resp. poly-IOP) into an IOPP (resp. IOP).

Theorem 8.1. *Let relation R be a relation that has an poly-IOPP with polynomials over field $\mathbb{F}$, maximal prover message degree d_{max} where the prover queries at most $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}}$ functions and let $0 < \rho < 1$ be a parameter.*

If $\mathbb{F}^*$ contains a multiplicative subgroup G whose order is a power of two and $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot \frac{d_{\mathsf{max}}+1}{\rho}$ then there is an IOPP for R with the following parameters:

	poly-IOPP for R	$\rightarrow$ IOPP for R
Proximity error	β_{PIOP}	$\max\left\{\beta_{\mathsf{PIOP}},\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{d^2\cdot(1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)\right\}$
Rounds	$\mathsf{k}_{\mathsf{PIOP}}$	$2\mathsf{k}_{\mathsf{PIOP}} + O(\log\log d_{\mathsf{max}})$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{m}_{\mathsf{PIOP}}$ (polynomials)	$O\left(\frac{d_{\mathsf{max}}}{\rho}\cdot\left(\mathsf{m}_{\mathsf{PIOP}} + \frac{1}{\rho}\right) + \mathsf{q}_{\mathsf{PIOP},\pi}\right)$
Oracle input queries	$\mathsf{q}_{\mathsf{PIOP},\mathbb{w}}$	$\mathsf{q}_{\mathsf{PIOP},\mathbb{w}}$
Proof queries	$\mathsf{q}_{\mathsf{PIOP},\pi}$	$O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \log\log d_{\mathsf{max}})$
Randomness	$\mathsf{r}_{\mathsf{PIOP}}$	$\mathsf{r}_{\mathsf{PIOP}} + (\mathsf{k}_{\mathsf{PIOP}} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + O(\log\log d_{\mathsf{max}}))\cdot\log\lvert\mathbb{F}\rvert$
Verifier running time	$\mathsf{vt}_{\mathsf{PIOP}}$	$O(\mathsf{vt}_{\mathsf{PIOP}}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \sqrt{d_{\mathsf{max}}})$

Proof. Let $L \subseteq \mathbb{F}$ be a domain with $|L| = (d_{\mathsf{max}}+1)/\rho$, and let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d_{\mathsf{max}}]$ be a Reed–Solomon code whose rate is ρ. We apply Theorem 5.1 with the following ingredients:

- The poly-IOPP for R given in the theorem statement.
- The IOPP for $\mathcal{C}$ given by Theorem 6.1. Observe that $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot |L|$.
- The proximity generator given by Item 2 of Theorem 4.2 for $2\mathsf{q}_{\mathsf{PIOP},\mathsf{m}}$ functions with proximity bound $\sqrt{\rho}$, seed length $2\mathsf{q}_{\mathsf{PIOP},\mathsf{m}}\cdot\log|\mathbb{F}|$, error $O\left(\frac{d_{\mathsf{max}}^2\cdot(1/\rho)^4}{|\mathbb{F}|}\right)$ and computation time $O(\mathsf{q}_{\mathsf{PIOP},\mathsf{m}})$.
- $\gamma := 1 - 2\sqrt{\rho}$.

The resulting IOPP for R has the following parameters:

- ***Proximity error.*** The proximity error of the IOPP for $\mathcal{C}$, when $\delta = 1 - 2\sqrt{\rho}$ is

$$\beta_{\mathsf{prx}}(1 - 2\sqrt{\rho}) = \max\left\{2\rho^{1/2},\ \rho^{1/4} + O\left(\frac{d_{\mathsf{max}}^2\cdot(1/\rho)^4}{|\mathbb{F}|}\right)\right\} \ .$$

The proximity error of the resulting IOPP is therefore at most

$$\begin{aligned}&\max\left\{\beta_{\mathsf{PIOP}}, \beta_{\mathsf{prx}}(1-2\sqrt{\rho}) + O\left(\frac{d_{\mathsf{prx}}^2 \cdot (1/\rho_{\mathsf{prx}})^2}{|\mathbb{F}|}\right)\right\}\\ &= \max\left\{\beta_{\mathsf{PIOP}},\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{d^2 \cdot (1/\rho)^4}{|\mathbb{F}|}\right)\right\}\ .\end{aligned}$$

- ***Rounds.*** The number of rounds is $2\mathsf{k}_{\mathsf{PIOP}} + O(\log\log d_{\mathsf{max}}) + 1 = 2\mathsf{k}_{\mathsf{PIOP}} + O(\log\log d_{\mathsf{max}})$.
- ***Proof length.*** The proof length is $O(\mathsf{m}_{\mathsf{PIOP}} \cdot |L| + \mathsf{q}_{\mathsf{PIOP},\pi} + |L|/\rho) = O\left(\frac{d_{\mathsf{max}}}{\rho} \cdot \left(\mathsf{m}_{\mathsf{PIOP}} + \frac{1}{\rho}\right) + \mathsf{q}_{\mathsf{PIOP},\pi}\right)$.
- ***Oracle input queries.*** The verifier makes $\mathsf{q}_{\mathsf{PIOP},\mathbb{w}}$ queries to its input.
- ***Proof queries.*** The proof query complexity is $O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \log\log d_{\mathsf{max}})$.
- ***Randomness.*** The randomness complexity is $\mathsf{r}_{\mathsf{PIOP}} + (\mathsf{k}_{\mathsf{PIOP}} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + O(\log\log d_{\mathsf{max}})) \cdot \log|\mathbb{F}|$.
- ***Verifier running time.*** The verifier running time is $O(\mathsf{vt}_{\mathsf{PIOP}} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \mathsf{k}_{\mathsf{PIOP}}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\pi} + \sqrt{d_{\mathsf{max}}}) = O(\mathsf{vt}_{\mathsf{PIOP}}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \sqrt{d_{\mathsf{max}}})$.

□

8.2 IOPPs for RS codes over every domain

In this section we show that if there is a proximity test for $\mathsf{RS}[\mathbb{F}, L_*, d_*]$ for a specific domain, then there is a proximity test for $\mathsf{RS}[\mathbb{F}, L, d]$ for every evaluation domain L_* and $d \leq d_*$.

Theorem 8.2. *Consider the following ingredients:*

- *A Reed–Solomon code* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$.
- *An IOPP* $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *for* $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}[\mathbb{F}, L_{\mathsf{prx}}, d_{\mathsf{prx}}]$. *with rate* $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}} + 1)/|L_{\mathsf{prx}}|$.
- Gen *is proximity generator for* $\mathcal{C}_{\mathsf{prx}}$, 2 *functions, seed length* $\mathsf{w}_{\mathsf{gen}}$, *proximity bound* ψ_{gen}, *and error* $\varepsilon_{\mathsf{gen}}$.

If $d \leq d_{\mathsf{prx}}$, $0 < \gamma < 1 - \max\{\psi_{\mathsf{gen}}, 2\rho_{\mathsf{prx}}\}$ *and* $\mathcal{C}_{\mathsf{prx}}$ *is* (γ, ℓ)*–list decodable, then there is an IOPP for* $\mathcal{C}$ *with the following parameters:*

	IOPP for $\mathcal{C}_{\mathsf{prx}}$	$\to$ IOPP for $\mathcal{C}$
Proximity error	β_{prx}	$\max\left\{1-\delta, \beta_{\mathsf{prx}} + \varepsilon_{\mathsf{gen}} + d_{\mathsf{prx}} \cdot \ell^2/\lvert\mathbb{F}\rvert\right\}$
Rounds	$\mathsf{k}_{\mathsf{prx}}$	$\mathsf{k}_{\mathsf{prx}} + 3$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert L_{\mathsf{prx}}\rvert) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{prx},f}$	1
Proof queries	$\mathsf{q}_{\mathsf{prx},\pi}$	$O(\mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\pi}$
Randomness	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{prx}} + \mathsf{w}_{\mathsf{gen}} + 2\log\lvert\mathbb{F}\rvert$
Verifier running time	$\mathsf{vt}_{\mathsf{prx}}$	$\mathsf{vt}_{\mathsf{prx}} + \mathsf{t}_{\mathsf{gen}} + O(1)$

Above, $\varepsilon_{\mathsf{gen}} := \varepsilon_{\mathsf{gen}}(\gamma)$ *and* $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$.

Proof. We first present a poly-IOPP for $\mathcal{C}$ and then apply Theorem 5.1 with this poly-IOPP, using $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ as the Reed–Solomon proximity test. Details follow.

The poly-IOPP is as follows: given a function $f\colon L \to \mathbb{F}$,

1. $\mathbf{P}_{\mathsf{PIOP}}$: compute and send $\hat{f} \in \mathbb{F}^{\leq d}[X]$, the extension of f to a degree d polynomial.
2. $\mathbf{V}$: sample a $a \leftarrow \mathbb{F}$ uniformly at random and accept if and only if $f(a) = \hat{f}(a)$.

We analyze the parameters of the poly-IOPP described above:

- *Completeness.* If $f \in \mathcal{C} := \mathsf{RS}[\mathbb{F}, L, d]$ then the interpolation done by $\mathbf{P}_{\mathsf{PIOP}}$ will be successful, and, by definition, $f(a) = \hat{f}(a)$ for every $a \in L$. Therefore, $\mathbf{V}_{\mathsf{PIOP}}$ will accept with probability 1.
- *Proximity.* Consider a function f with $\Delta(f, \mathcal{C}) \geq \delta$. By definition, for every $\hat{g} \in \mathbb{F}^{\leq d}[X]$ that could be sent by the prover, we have that $\hat{g}$ and f agree on at most $1 - \delta$ of the points of L. Hence, $f(a) = \hat{g}(a)$ with probability at most $1 - \delta$, and so the verifier accepts with probability at most $1 - \delta$.
- *Message degrees.* The prover sends a single polynomial of degree at most d.
- *Complexity parameters.* Randomness complexity $\log |L|$, input-query complexity 1, proof-query complexity 1, and verifier running time $O(1)$ field operations.

We now apply Theorem 5.1 with $d_{\mathsf{max}} := d_{\mathsf{prx}}$ and γ to the poly-IOPP described above using the IOPP $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ and the proximity generator Gen. It holds that $d \leq d_{\mathsf{prx}} = d_{\mathsf{max}}$, that $0 < \gamma < 1 - \max\{\psi_{\mathsf{gen}}, 2\rho_{\mathsf{prx}}\}$, and, that $\mathcal{C}_{\mathsf{prx}}$ is (γ, ℓ)–list decodable. We can therefore apply Theorem 5.1. The parameters of the resulting IOPP follow. □

8.3 Testing bivariate RM codes with inverse polynomial error

In this section we show a proximity test for (individual-degree) bivariate Reed–Muller codes:

Theorem 8.3. *Let $\mathbb{F}$ be a field, G be a multiplicative subgroup of $\mathbb{F}^*$ whose order is a power of two, and $D \subseteq \mathbb{F} \times \mathbb{F}$ be an admissible domain with row-index domain L_X (as per Definition 6.5). If $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^{12} \cdot (d_{\mathsf{max}} + 1)/\rho_X$ then the bivariate Reed–Muller code $\mathcal{C} := \mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$ has an IOPP with the following parameters:*

IOPP to show δ proximity to $\mathcal{C}$	
Proximity error	$\max\left\{2\sqrt{1-\delta},\ \rho_X^{1/4}\right\} + O\left(\frac{d_{\mathsf{max}}^2 \cdot (1/\rho_X)^4}{\lvert\mathbb{F}\rvert}\right)$
Alphabet	$\mathbb{F}$
Rounds	$O(\log\log d_{\mathsf{max}})$
Proof length	$O\left(\frac{d_{\mathsf{max}} \cdot \lvert L_X\rvert}{\rho_X}\right)$
Oracle input queries	1
Proof queries	$O(\log\log d_{\mathsf{max}})$
Randomness	$O(\log\log d_{\mathsf{max}} \cdot \log \lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(d_Y + \sqrt{d_{\mathsf{max}}})$

Above, $d_{\mathsf{max}} := \max\{d_X, d_Y\}$ and $\rho_X := (d_X + 1)/|L_X|$.

Proof. We begin by instantiating a poly-IOPP for the code $\mathcal{C}$ using Lemma 6.6 with the following ingredients:

- The Reed–Muller code $\mathcal{C} := \mathsf{RM}[\mathbb{F}, D, (d_X, d_Y)]$.
- $H \subseteq G$ is a multiplicative subgroup of $\mathbb{F}^*$ whose order is a power of two with $d_Y \leq |H| < 2d_Y$.
- Gen is the strong proximity generator for $\mathcal{C}_X := \mathsf{RS}[\mathbb{F}, L_X, d_X]$ described in Item 1 of Theorem 4.4 for $|H|$ functions with seed length $\log |\mathbb{F}|$, proximity bound $\sqrt{\rho_X}$, and error $O\left(\frac{d_X^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right)$. The computation time of computing the interpolation polynomial of the coefficients output by the proximity generator is $\tilde{O}(|H|) = \tilde{O}(d_Y)$.

Notice that $|\mathbb{F}| \geq |L_X|^2$ and $|H| \geq d_Y$, and so the requirements for Lemma 6.6 have been met. The resulting poly-IOPP for $\mathcal{C}$ has the following parameters:

poly-IOPP to show $\delta < 1 - \rho_X$ proximity to $\mathcal{C}$	
Proximity error	$2\sqrt{1-\delta} + O\left(\frac{d_Y + d_X^2 \cdot (1/\rho_X)^4}{\lvert\mathbb{F}\rvert}\right)$
Rounds	3
Alphabet	$\mathbb{F}$
Number of polynomials	$\lvert L_X\rvert + 3$
Oracle input queries	1
Proof queries	5
Randomness	$4 \log \lvert\mathbb{F}\rvert$
Verifier running time	$\tilde{O}(d_Y)$
Max message degree	$\max\{d_X, 2d_Y - 1\}$

Notice that the query complexity of the poly-IOPP is 5, and so the number of polynomials queried by the verifier is at most $\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} := 5$.

We now compile the above poly-IOPP for $\mathcal{C}$ into an IOPP for $\mathcal{C}$ using Theorem 8.1, with $\rho := \frac{1}{16} \cdot \rho_X$. Observe that $\mathbb{F}^*$ contains a multiplicative subgroup G whose order is a power of two and $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot \frac{d_{\mathsf{max}}+1}{\rho}$, and so the requirements for Theorem 8.1 have been met. The resulting IOPP has the following parameters:

- ***Proximity error.*** The proximity error of the resulting IOPP is therefore at most

$$\begin{aligned}
&\max\left\{\beta_{\mathsf{PIOP}},\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{d_{\mathsf{max}}^2 \cdot (1/\rho)^4}{|\mathbb{F}|}\right)\right\} \\
&\leq \max\left\{\beta_{\mathsf{PIOP}},\ \rho_X^{1/4} + O\left(\frac{d_{\mathsf{max}}^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right)\right\} \\
&= \max\left\{2\sqrt{1-\delta} + O\left(\frac{d_Y + d_X^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right),\ \rho_X^{1/4} + O\left(\frac{d_{\mathsf{max}}^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right)\right\} \\
&\leq \max\left\{2\sqrt{1-\delta},\ \rho_X^{1/4}\right\} + O\left(\frac{d_{\mathsf{max}}^2 \cdot (1/\rho_X)^4}{|\mathbb{F}|}\right)\ ,
\end{aligned}$$

  where the first equality follows since $\rho := \frac{1}{16} \cdot \rho_X$, and so $2\rho^{1/2} < \rho^4 < \rho_X^{1/4}$. Observe that, while the poly-IOPP soundness error holds only when $\delta < 1 - \rho_X$, the resulting soundness error is greater than ρ_X, and so we can remove this requirement.

- ***Rounds.*** The number of rounds is $10 + O(\log\log d_{\mathsf{max}}) = O(\log\log d_{\mathsf{max}})$.

- ***Proof length.*** The proof length is $O\left(\frac{d_{\mathsf{max}}}{\rho} \cdot \left(|L_X| + \frac{1}{\rho}\right) + 5\right) = O\left(\frac{d_{\mathsf{max}} \cdot |L_X|}{\rho_X}\right)$.

- ***Oracle input queries.*** The verifier makes 1 queries to its input.

- ***Proof queries.*** The proof query complexity is $O(5 + 5 + \log\log d_{\mathsf{max}}) = O(\log\log d_{\mathsf{max}})$.

- ***Randomness.*** The randomness complexity is $4 \log |\mathbb{F}| + (3 + 5 + O(\log\log d_{\mathsf{max}})) \cdot \log |\mathbb{F}| = O(\log\log d_{\mathsf{max}} \cdot \log |\mathbb{F}|)$.

- ***Verifier running time.*** The verifier running time is $\tilde{O}(d_Y) + \tilde{O}(5 + \sqrt{d_{\mathsf{max}}}) = \tilde{O}(d_Y + \sqrt{d_{\mathsf{max}}})$.

	poly-IOPP for R	*$\to$ IOPP for R*
Proximity error	β_{PIOP}	$\max\left\{\beta_{\mathsf{PIOP}},\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{d^2\cdot(1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)\right\}$
Rounds	$\mathsf{k}_{\mathsf{PIOP}}$	$2\mathsf{k}_{\mathsf{PIOP}} + O(\log\log d_{\mathsf{max}})$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{m}_{\mathsf{PIOP}}$ *(polynomials)*	$O\left(\frac{d_{\mathsf{max}}}{\rho}\cdot\left(\mathsf{m}_{\mathsf{PIOP}} + \frac{1}{\rho}\right) + \mathsf{q}_{\mathsf{PIOP},\pi}\right)$
Oracle input queries	$\mathsf{q}_{\mathsf{PIOP},\mathsf{w}}$	$\mathsf{q}_{\mathsf{PIOP},\mathsf{w}}$
Proof queries	$\mathsf{q}_{\mathsf{PIOP},\pi}$	$O(\mathsf{q}_{\mathsf{PIOP},\pi} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \log\log d_{\mathsf{max}})$
Randomness	$\mathsf{r}_{\mathsf{PIOP}}$	$\mathsf{r}_{\mathsf{PIOP}} + (\mathsf{k}_{\mathsf{PIOP}} + \mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + O(\log\log d_{\mathsf{max}}))\cdot\log\lvert\mathbb{F}\rvert$
Verifier running time	$\mathsf{vt}_{\mathsf{PIOP}}$	$O(\mathsf{vt}_{\mathsf{PIOP}}) + \tilde{O}(\mathsf{q}_{\mathsf{PIOP},\mathsf{m}} + \sqrt{d_{\mathsf{max}}})$

□

Acknowledgments

Gal Arnon is supported in part by a grant from the Israel Science Foundation (Grant No. 2686/20), by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness, and by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center. Alessandro Chiesa is supported in part by the Ethereum Foundation. Eylon Yogev is supported by an Alon Young Faculty Fellowship, by the Israel Science Foundation (Grant No. 2893/22), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

[ABCY22] Gal Arnon, Amey Bhangale, Alessandro Chiesa, and Eylon Yogev. "A Toolbox for Barriers on Interactive Oracle Proofs". In: *Proceedings of the 20th Theory of Cryptography Conference*. TCC '22. 2022, pp. 447–466.

[ACY22a] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "A PCP Theorem for Interactive Proofs". In: *Proceedings of the 41st Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '22. 2022, pp. 64–94.

[ACY22b] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "Hardness of Approximation for Stochastic Problems via Interactive Oracle Proofs". In: *Proceedings of the 37th Annual IEEE Conference on Computational Complexity*. CCC '22. 2022, 24:1–24:16.

[ALMSS98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (1998). Preliminary version in FOCS '92., pp. 501–555.

[AS03] Sanjeev Arora and Madhu Sudan. "Improved Low-Degree Testing and its Applications". In: *Combinatorica* 23.3 (2003). Preliminary version appeared in STOC '97., pp. 365–426.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic checking of proofs: a new characterization of NP". In: *Journal of the ACM* 45.1 (1998). Preliminary version in FOCS '92., pp. 70–122.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 19–46.

[BCGGHJ17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. ASIACRYPT '17. 2017, pp. 336–365.

[BCGRS17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Interactive Oracle Proofs with Constant Rate and Query Complexity". In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*. ICALP '17. 2017, 40:1–40:15.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. "Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs". In: *Proceedings of the 13th Theory of Cryptography Conference*. TCC '16-A. 2016, pp. 33–64.

[BCIKS20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. "Proximity Gaps for Reed–Solomon Codes". In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. FOCS '20. 2020, pp. 900–909.

[BCL22] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. "Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier". In: *Proceedings of the 41st Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '22. 2022, pp. 275–304.

[BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. 2019, pp. 103–128.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking computations in polylogarithmic time". In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. STOC '91. 1991, pp. 21–32.

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

[BGHSV06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. "Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding". In: *SIAM Journal on Computing* 36.4 (2006), pp. 889–974.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. "DEEP-FRI: Sampling Outside the Box Improves Soundness". In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. ITCS '20. 2020, 5:1–5:32.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. "Efficient Probabilistically Checkable Proofs and Applications to Approximations". In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*. STOC '93. 1993, pp. 294–304.

[BN22] Sarah Bordage and Jade Nardi. "Interactive Oracle Proofs of Proximity to Algebraic Geometry Codes". In: *Proceedings of the 37th Annual IEEE Conference on Computational Complexity*. CCC '22. 2022, 30:1–30:45.

[BS06] Eli Ben-Sasson and Madhu Sudan. "Robust locally testable codes and products of codes". In: *Random Structures and Algorithms* 28.4 (2006), pp. 387–402.

[BS08] Eli Ben-Sasson and Madhu Sudan. "Short PCPs with Polylog Query Complexity". In: *SIAM Journal on Computing* 38.2 (2008). Preliminary version appeared in STOC '05., pp. 551–607.

[Bab85] László Babai. "Trading group theory for randomness". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. STOC '85. 1985, pp. 421–429.

[Ben+17] Eli Ben-Sasson et al. "Computational integrity with a public random string from quasi-linear PCPs". In: *Proceedings of the 36th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '17. 2017, pp. 551–579.

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

[CMS20] Alessandro Chiesa, Peter Manohar, and Igor Shinkar. "On Axis-Parallel Tests for Tensor Product Codes". In: *Theory of Computing* 16 (2020), pp. 1–34.

[CY21a] Alessandro Chiesa and Eylon Yogev. "Subquadratic SNARGs in the Random Oracle Model". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 711–741.

[CY21b] Alessandro Chiesa and Eylon Yogev. "Tight Security Bounds for Micali's SNARGs". In: *Proceedings of the 19th Theory of Cryptography Conference*. TCC '21. 2021, pp. 401–434.

[DFKRS11] Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. "PCP Characterizations of NP: Toward a Polynomially-Small Error-Probability". In: *Computational Complexity* 20.3 (2011), pp. 413–504.

[DHK15] Irit Dinur, Prahladh Harsha, and Guy Kindler. "Polynomially Low Error PCPs with polyloglog n Queries via Modular Composition". In: *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*. STOC '15. 2015, pp. 267–276.

[DS14] Irit Dinur and David Steurer. "Analytical approach to parallel repetition". In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. STOC '14. 2014, pp. 624–633.

[Din07] Irit Dinur. "The PCP theorem by gap amplification". In: *Journal of the ACM* 54.3 (2007), p. 12.

[FGLSS96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. "Interactive proofs and the hardness of approximating cliques". In: *Journal of the ACM* 43.2 (1996). Preliminary version in FOCS '91., pp. 268–292.

[GLSTW23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. "Brakedown: Linear-time and field-agnostic SNARKs for R1CS". In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO '23. 2023.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof systems". In: *SIAM Journal on Computing* 18.1 (1989). Preliminary version appeared in STOC '85., pp. 186–208.

[KR08] Yael Kalai and Ran Raz. "Interactive PCP". In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*. ICALP '08. 2008, pp. 536–547.

[MR08] Dana Moshkovitz and Ran Raz. "Two-query PCP with subconstant error". In: *Journal of the ACM* 57 (5 2008). Preliminary version appeared in FOCS '08., pp. 1–29.

[MR10] Dana Moshkovitz and Ran Raz. "Sub-Constant Error Probabilistically Checkable Proof of Almost-Linear Size". In: *Computational Complexity* 19.3 (2010), pp. 367–422.

[Mie09] Thilo Mie. "Short PCPPs verifiable in polylogarithmic time with O(1) queries". In: *Annals of Mathematics and Artificial Intelligence* 56 (3 2009), pp. 313–338.

[Mos19] Dana Moshkovitz. *Sliding Scale Conjectures in PCP*. 2019.

[PS94] Alexander Polishchuk and Daniel A. Spielman. "Nearly-linear size holographic proofs". In: *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. STOC '94. 1994, pp. 194–203.

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

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

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

[RS97] Ran Raz and Shmuel Safra. "A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. STOC '97. 1997, pp. 475–484.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. "Interactive proofs of proximity: delegating computation in sublinear time". In: *Proceedings of the 45th ACM Symposium on the Theory of Computing*. STOC '13. 2013, pp. 793–802.

[Raz95] Ran Raz. "A parallel repetition theorem". In: *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*. STOC '95. 1995, pp. 447–456.

[XZZPS19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.