עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

מאת
גל ארנון

By
Gal Arnon

התקדמויות חדשות בהוכחות אורקל אינטראקטיביות: תיאוריה, פרקטיקה, ומגבלות

New Advancements in Interactive Oracle Proofs: Theory, Practice, and Limitations

מנחה:
פרופ' מוני נאור
ד"ר אילון יוגב

Advisor:
Prof. Moni Naor
Dr. Eylon Yogev

אייר תשפ"ה

May 2025

New Advancements in Interactive Oracle Proofs: Theory, Practice, and Limitations

Gal Arnon

Under the Supervision of Prof. Moni Naor (Weizmann Institute) and
Dr. Eylon Yogev (Bar-Ilan University)

Department of Computer Science and Applied Mathematics

Weizmann Institute of Science

Submitted for the degree of Doctor of Philosophy

to the Scientific Council of the Weizmann Institute of Science

May 2025

Dedicated to my grandparents,
Yehoshua, Zipporah, Michal, and Michael.

Abstract

Probabilistic proof systems allow a powerful prover to convince a computationally weak verifier of the validity of a large and complex computation. These seemingly magical objects have been an extremely important tool both in theory, where they have driven breakthroughs like the PCP theorem, zero-knowledge proofs, and advancements in approximation hardness, and in practice, where they are integral to the scalability of cloud computing and blockchain technology and are widely deployed, securing billions of dollars worth of transactions.

We focus on interactive oracle proofs (IOPs), a probabilistic proof model where the prover and verifier interact over a number of rounds, and after the interaction, the verifier probabilistically reads a small number of bits from each prover message and decides to accept or reject based on the examined locations. IOPs are incredibly powerful tools: theoretically, they allow for achieving efficiency parameters beyond what is known for other probabilistic proof systems, and practically, efficient IOPs can be compiled into incredibly fast and succinct cryptographic proofs that have found wide adoption toward securing real-world systems.

In this thesis, we further the understanding of the theory, practice, and limitations of IOPs and related proof systems by: (1) providing an analog of the PCP theorem for IOPs by establishing that small-query IOPs are as computationally powerful as interactive proofs, (2) constructing new IOPs for NP with small soundness error and low query complexity, (3) developing new concretely efficient IOPs of proximity for Reed–Solomon codes, and (4) demonstrating barriers for constructing efficient IOPs and PCPs.

Declaration

I hereby declare that this thesis summarizes my original research, performed under the guidance of my advisors Moni Naor and Eylon Yogev. Chapters 3 and 4 are based on work written in collaboration with Alessandro Chiesa and Eylon Yogev. Chapter 5 is based on research written in collaboration with Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. Chapter 6 is based on research written in collaboration with Amey Bhangale, Alessandro Chiesa, and Eylon Yogev.

Acknowledgements

When starting a Ph.D., one naturally prepares for a marathon, with many ups and downs, back and forths, failures and (hopefully) successes. Starting this journey, I could not have guessed that my studies would instead be a beacon of stability amid a chaotic and tumultuous world. I hope to look back upon this passage in the future and see that the world is a better place than during the past few years.

First and foremost, I would like to thank my advisors, Moni Naor and Eylon Yogev, for their unwavering support and advice and for guiding me toward completing my degree, culminating in this thesis. In particular, I see my success as the direct result of my amazing relationship with Eylon. It is impossible to summarize all the long hours discussing life and research until we either realize how late it is or the phone line cuts because we talked for too long (if anyone was wondering - it cuts off after 2 hours), so I will not even attempt it. So long, and thanks for all the coffee.

I would also like to extend a special thanks to my unofficial third advisor, Alessandro Chiesa, for his advice on writing and research, for help throughout my academic career, and for pushing me out of my comfort zone.

I want to extend my gratitude to all of the amazing coauthors I had along the way, without whom none of this would have been possible: Amey Bhangale, Shany Ben-David, Alessandro Chiesa, Giacomo Fenzi, Tomer Grossman, Guy Rothblum, and Eylon Yogev. Additionally, I would like to thank all of my colleagues and friends at the Weizmann Institute, at the Cyber Center in Bar-Ilan University, and at the Computation Security Lab at EPFL (especially Ziyi Guan who has been explicitly mentioned at her insistence).

Additional thanks go out to my Ph.D. supervisory committee, Guy Rothblum, and Irit Dinur, for the advice and for making sure that I am focused on the right goals.

Finally, I would like to thank my family and close friends for keeping me (relatively) sane during this time and for your infinite supply of love and support over the years.

– Gal Arnon

Contents

Chapter 1

Introduction

Probabilistic proofs play a central role in complexity theory and cryptography. In the past decades, probabilistic proofs have become powerful and versatile tools in these fields, leading to breakthroughs in zero-knowledge, delegation of computation, hardness of approximation, and other areas.

A brief history of proof systems. The history of research into probabilistic proof systems in their modern form begins with the introduction of interactive proofs (IPs) [BM88; GMR89]. In an IP, an all-powerful prover interacts over multiple rounds with a computationally weak (polynomial-time) probabilistic verifier and aims to convince it of the validity of some statement. It is this combination of interaction and randomness which seemingly confers IPs much more power than their deterministic and non-interactive counterpart in NP proof systems. Indeed, the power of IPs has been shown to be significantly higher than what is believed for non-deterministic polynomial-time computation, culminating in the celebrated IP = PSPACE theorem [LFKN92; Sha92]. Furthermore, IPs have found significant uses in cryptography due to the ability to make them have a zero-knowledge property [GMR89; GMW91], informally meaning that no information is leaked to the verifier beyond the validity of the statement.

The development of IPs, in turn, led to probabilistically checkable proofs (PCPs) [BFLS91; FGLSS96], where a probabilistic polynomial-time verifier has query access to a proof string, and there is no further interaction. A line of works culminated in the PCP Theorem [AS98; ALMSS98], which can be stated as; every language in NP can be decided, with constant soundness error, by probabilistically examining only a constant number of bits in a polynomially long proof. The PCP theorem can also be viewed as establishing tight NP-hardness results for deciding even the approximate versions of many NP-complete problems. These and other advances in probabilistic proofs have reshaped theoretical computer science.

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 [BGHSV05], and achieving PCPs with linear proof length remains an open problem.

Interactive oracle proofs. In order to circumvent the limitations of PCPs, researchers

formulated *interactive oracle proofs* (IOPs) [BCS16; RRR16], which combines aspects of the IPs and PCPs (IOPs further generalize the notion of *interactive* PCPs first formulated by [KR08]). A k-round IOP can be viewed as a k-round IP where the verifier has PCP-like access to each prover message: the prover and verifier interact for k rounds, and after the interaction the verifier probabilistically reads a small number of bits from each prover message and decides to accept or reject based on the examined locations.

Adding interaction turns out to have a dramatic effect on the efficiency of IOPs when compared to PCPs. For instance, most efficient PCPs use a technique called proof composition to reduce query complexity at the cost of a blowup in proof length. In an IOP, we can apply proof composition interactively, where the prover can supply the verifier with only a small fraction of what it would have had to send in the PCP. As a result, not long after the introduction of IOPs, it was shown that there exist IOPs for NP with linear length [BCGRS17]. Thanks to this and many other techniques, known IOPs now 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; ACFY24]. Another line of work shows that IOPs can also be used to prove hardness of approximation results for certain stochastic problems [CFLS97; Dru11a; ACY22a; ACY22b].

Succinct cryptographic proofs. A succinct non-interactive argument (SNARG) can be seen as a cryptographic analog of an NP proof system: as in an NP proof, the prover sends a single message to the verifier, who reads it and makes its decision about the statement being proven. However, in a SNARG, cryptographic tools are leveraged so that this message is much shorter than the information-theoretic NP witness.

The connection between probabilistic proofs and SNARGs is a deep one, as first shown in [Mic00]. First, the transformation of Kilian [Kil92] is used to construct succinct *interactive* arguments: the prover succinctly commits to the PCP string in a way that enables opening individual symbols from the string, and the verifier then chooses its queries and sends them to the prover, who opens the relevant locations for the verifier. Next, the Fiat–Shamir [FS86] heuristic is used in order to make the argument non-interactive. This is done by using a hash function to mimic the verifier's messages. If this hash function has strong enough security properties, then this leads to a SNARG. By inspecting the construction, it can be seen that shorter PCPs with smaller query complexity lead to smaller commitments and fewer openings in the succinct argument, and so to shorter SNARGs. Unfortunately, due to our current inability to construct concretely efficient PCPs, their use in constructing SNARGs has been limited.

Thus, one of the main rationales for the introduction of IOPs was to construct more efficient SNARGs. Indeed, [BCS16] show that the transformation described above holds also for IOPs, with shorter and more query-efficient IOPs yielding shorter SNARGs. It was later shown that in some models, this connection is inherent, as effi-

cient SNARGs imply efficient IOPs [RV09; CY20]. Thus, the development of efficient IOPs has gone hand-in-hand with new SNARGs, which have, over time, become practically efficient and now underlie highly efficient cryptographic proofs that have seen widespread deployment in real-world applications. Indeed, IOPs underly numerous SNARG-based real-world systems, [Pol; Ris; Stab; Staa; Zks; San; Mid; Nep; Ola; Her]. Perhaps the most well-known real-world use of IOP-based SNARGs is in blockchain systems, where they secure billions of dollars worth of transactions.

1.1 The computational power of IOPs

One of the main contributions of computational complexity is the introduction of various computational models and the analysis of their computational capacity. Analyzing the computational power of different models allows us to understand what it means to solve a problem and find barriers to constructing algorithms that solve these problems. For example, these formulations have allowed posing the famous P vs NP question the answer to which would have wide significance to all of science. Thus, with the introduction of probabilistic proof systems, much work was done to analyze what can and cannot be decided by them.

Significant research led to the full characterization of interactive proofs, culminating in the IP = PSPACE theorem [LFKN92; Sha92]. For PCPs, The breakthrough "PCP Theorem" [AS98; ALMSS98] shows that every language in NP has a PCP with polynomial length and constant query complexity and alphabet sizes. It is one of the most important results in theoretical computer science with applications to cryptography and hardness of approximation, to name just a few. It is, therefore, natural to ask:

What is the computational power of efficient IOPs?

Most research regarding IOPs has focused on understanding IOPs for languages in NP (and more generally, various forms of non-deterministic computations) while using the additional rounds of interaction to achieve better efficiency compared to PCPs for those languages. While much attention has been given to NP, relatively little was known beyond NP; prior works either targeted classifying IOPs relative to IPs with a single round [Dru11a] or went all the way up to PSPACE [CFLS95; CFLS97].

Charecterizing IOPs. In Chapter 3, which is based on [ACY22a; ACY22b], we prove an analog of the PCP theorem for all languages that have interactive proofs and give new hardness of approximation results for stochastic problems. Surprisingly, we show that any k-round IP can be transformed into a k-round IOP where the verifier makes $o(k)$ bit queries. We extend this result by showing how to transform both the sumcheck protocol [LFKN92] and Shamir's protocol [Sha92] into IOPs where the verifier makes a constant number of bit queries. This establishes that PSPACE has an IOP where the verifier queries $O(1)$ bits.

Observe that the IOPs we construct include rounds of interaction that are never queried by the verifier. This is surprising in light of [GVW02], which shows that IOPs with k rounds capture more languages than IOPs with $o(k)$ rounds (under mild complexity assumptions). In other words, our result shows, almost paradoxically, that while round complexity is crucial to the descriptive power of IOPs, reading all of these rounds is not essential.

1.2 Efficient IOPs from proofs of proximity

The construction of efficient IOPs (and, indeed, PCPs) tends to follow a blueprint: an incredibly efficient IOP is designed in an idealized model, where malicious provers are restricted to act in certain ways, and this idealized IOP is then compiled into a standard IOP via additional protocols testing that the prover acted according to the restricted model. The most common idealized model are "polynomial IOPs", where the prover can only send low-degree polynomials. The compilation to a standard IOP, then, reduces to testing that the prover sent a string that is close to conforming to the evaluation of a low-degree polynomial. The family of strings that agree with the evaluations of (univariate) low-degree polynomials is known as the Reed–Solomon (RS) code.

Thus, designing efficient IOPs reduces to designing efficient IOPs *of proximity* (IOPPs) for RS codes. In more detail in an IOPP for RS codes, the verifier has oracle access to a function f and is asked to distinguish, with high probability and small query complexity, between the case that f is the evaluation of a degree d univariate polynomial and the case that f is far (in Hamming distance) from the evaluation of any such polynomial. The verifier is aided in this task by an untrusted but prover (with IOP-like communication) with full access to f. Proximity tests for RS codes in various models have received a lot of attention, focusing on reducing the soundness error and query complexity [Din07; BS08; Mie09; RVW13; BBHR18; BGHSV06a; BGKS20; BCIKS20].

IOPs with inverse polynomial soundness error. A major open problem for PCPs and IOPs is the *sliding-scale conjecture* (in the inverse-polynomial error regime), which postulates a proof system for NP with constant query complexity and high soundness: it requires that every language in NP has a PCP/IOP with soundness error $1/\mathrm{poly}(n)$, proof length $\mathrm{poly}(n)$ over an alphabet of size $\mathrm{poly}(n)$, and query complexity $O(1)$ [BGLR93]. Proving this conjecture true would be a game changer for both practical systems and theory [Mos19].

In Chapter 4, which is based on [ACY23], we design an IOP of proximity for RS codes with query complexity $\log\log d$ and soundness error (roughly) $1 - O(\sqrt{\rho})$ where ρ is the rate of the RS code (in this case, the rate is $d/|\mathcal{L}|$, with $\mathcal{L}$ being the evaluation domain of the RS code). By utilizing this new proximity test and an efficient compilation procedure, we obtain the state-of-the-art for low-query IOPs with inverse polynomial soundness error. The construction achieves the soundness

error desired for the sliding-scale conjecture with the verifier making only $O(\log\log n)$ queries, a significant improvement over the prior best, which requires $\text{polyloglog}\, n$ queries [DHK15].

Concrete efficiency via improved RS proximity testing. In order to construct practically efficient IOP-based SNARGs, we must design IOPs with very small prover and verifier complexity. This, again, boils down to the design of very efficient IOPs of proximity. In Chapter 5, in line with this strategy, we develop the STIR protocol, which constitutes both a theoretical and practical breakthrough in RS proximity testing. To achieve soundness error $2^{-\lambda}$, protocol has query complexity (roughly) $O(\lambda \cdot \log\log d + \log d)$ compared to the previous best which required $O(\lambda \cdot \log d)$ queries [BBHR18; BCIKS20]. Implementing STIR, we show that the reduced query complexity of STIR yields significant improvements in real-world parameters when compared to prior protocols. This, in turn, allows for the implementation of new, more efficient SNARGs, thereby pushing these useful cryptographic tools more and more into the mainstream. The contents of Chapter 5 are based on [ACFY24].

1.3 Barriers to constructing efficient IOPs

Limitations of interactive proofs are scarce and hard to come by but are crucial to allowing us to know where to place the goalposts for further research, steering us away from time-consuming, fruitless endeavors, and for understanding what can be expected in terms of practical viability of these objects.

Some information-theoretic barriers are known for constructing IPs [GH98; GVW02], mostly dealing with the case where the prover-to-verifier communication is short. For PCPs, bounds can be shown by utilizing their inherent connection with hardness of approximation. In more detail, since PCPs show that deciding some languages is as hard as approximating the value, by utilizing approximation algorithms (e.g., [Sch78; Zwi98; MNT22]), we can rule out some PCPs for these languages. Furthermore, [FS11] show that *succinct* PCPs with certain parameters imply that the polynomial hierarchy collapses.

Since IOPs were defined in order to bypass limitations of PCPs, it is natural to ask:

What are the limitations of IOPs, and how do they compare to PCPs?

For example: What trade-offs are there between round complexity, query complexity, and soundness error in IOPs? How small can the soundness error of an IOP be if we require constant query complexity but allow increasing the alphabet size (as in a sliding-scale PCP)? There are few known bounds for IOPs: [CY20] extend some of the results of [GH98] to IOPs, and [NR22] show limitations for succinct IOPs for circuit satisfiability, where the proof length is polynomial in the number of circuit inputs.

In Chapter 6, which is based on [ABCY22], we provide a toolbox of transformations for IOPs consisting of: (i) tools for length and round reduction, (ii) tools for improving completeness, and (iii) tools for derandomization. We then utilize this

toolbox to establish new barriers to constructing proof systems. We show the following:

- Low-error IOPs can be transformed into low-error PCPs. In other words, interaction can be used to construct low-error PCPs; alternatively, low-error IOPs are as hard to construct as low-error PCPs. This relates IOPs to PCPs in the regime of the sliding scale conjecture for inverse-polynomial soundness error.
- Under standard complexity assumptions, short IOPs for 3SAT have high soundness error. For example, any IOP for 3SAT with quasilinear length and polylogarithmic round and query complexity has soundness error $\omega(1/n)$.
- IOPs where the query complexity is smaller than the round complexity cannot have good soundness error (under standard assumptions). This provides soundness error limitations for transformations that result in IOPs with k rounds and $o(k)$ query complexity, such as those described in Chapter 3.
- Limitations of binary-alphabet constant-query IOPs: while it is folklore knowledge that under complexity assumptions, PCPs with a binary alphabet and query complexity either 2 or 3 cannot have good soundness, it was not known whether interaction can help in further reducing the soundness error. We show that this is unlikely if the number of rounds is not large.

We believe that our toolbox will prove useful to establish additional barriers beyond those described above.

1.4 Organization

This thesis is organized into six chapters. Chapter 1 provides introductory material intermixed with a high-level overview of our contributions. Chapter 2 includes preliminaries that are used throughout the rest of this thesis. Each subsequent chapter contains parts of different papers, and can be read independently. In Chapter 3, which is based on [ACY22a; ACY22b], we show how to transform interactive proofs into IOPs with small query complexity. Chapter 4 is based on [ACY23], and deals with constructing IOPs with inverse polynomial soundness error and small query complexity. In Chapter 5 we construct a new concretely efficient IOP for testing Reed–Solomon codes. This chapter is based on [ACFY24]. Finally, Chapter 6 deals with barriers to constructing efficient IOPs, and is based on [ABCY22].

Chapter 2

Preliminaries

In this chapter we present definitions and theorems which will be useful in order to understand and prove our results. While the majority of the preliminaries are in this section, a small number of definitions relevant only for Chapter 3 appear separately in Section 3.3. Throughout this thesis, we use the following notation unless explicitly stated otherwise:

- In some theorems we highlight important parameters with a yellow background.
- For a finite field $\mathbb{F}$, the set $\mathbb{F}^{<d}[X]$ denotes all univariate polynomials of degree smaller than d. Similarly, the set $\mathbb{F}^{\leq d}[X]$ is the set of all polynomials with degree at most d.
- The "hat" symbol over a function (e.g., $\hat{p}$) denotes that it is a polynomial.
- For two functions $f, g \colon \mathcal{L} \to \mathbb{F}$, $\Delta(f,g)$ is the fractional Hamming distance between f and g (the fraction of points in which they disagree). For a set $\mathcal{S} \subseteq \mathbb{F}^{\mathcal{L}}$,

$$\Delta(f,\mathcal{S}) := \min_{h \in \mathcal{S}} \Delta(f,h)\,.$$

  We say that f and g are δ-far if $\Delta(f,g) > \delta$, and if $\Delta(f,g) \leq \delta$ then the functions are δ-close. Similarly, the relative distance between two strings $x, y \in \Sigma^m$ is the relative distance between the functions $f, g \colon [m] \to \Sigma$ such that $f(i) = x_i$ and $g(i) = y_i$.
- For a set $\mathcal{L} \subseteq \mathbb{F}$ and $k \in \mathbb{N}$, $\mathcal{L}^k := \{x^k : x \in \mathcal{L}\}$.
- A set $\mathcal{L} \subseteq \mathbb{F}$ is *smooth* if it is a multiplicative coset of $\mathbb{F}^*$ whose order is a power of 2.
- For a ternary relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$, let $L(\mathcal{R}) = \{(\mathbb{x}, \mathbb{y}) \mid \exists \mathbb{w},\ (\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}\}$ be the language induced by $\mathcal{R}$.
- We sometimes consider proof systems for *multi-indexed* relations. A multi-indexed relation $\mathcal{R}$ is a set of tuples $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$ where $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}]$ are the indexes, $\mathbb{x}$ the instance, and $\mathbb{w}$ the witness.

In this thesis, we will be interested in interactive algorithms that have access to oracles. An algorithm $\mathbf{A}$ with oracle access to a function $\mathcal{O} \colon S \to T$, denoted $\mathbf{A}^{\mathcal{O}}$, is a (possibly probabilistic) algorithm (e.g., a RAM machine) which has the ability to

query $\mathcal{O}$ on any input $z \in S$ and receive the answer $y = \mathcal{O}(z)$ in unit cost. An interactive algorithm $\mathcal{A}$ is a stateful algorithm that "interacts" with another party over multiple rounds, by, in each round, being invoked on the interaction transcript thus far, and outputting a new message. For interactive algorithms **A** and **B**, we denote by $\langle \mathbf{A}(a), \mathbf{B}(b) \rangle(c)$ the random variable describing the output of **B** following the interaction between **A** and **B**, where **A** is given private input a, **B** is given private input b, and both parties are given joint input c.

2.1 Probabilistic proof systems

Probabilistic proof systems, in which complex mathematical claims can be efficiently proven to a computationally weak verifier, are fundamental to the theory of computer science and to cryptography. Many different notions and definitions exist to formally realize this high-level idea, each with its own strengths and drawbacks. In this thesis we will focus on one such definition called *interactive oracle proofs*, which are useful both as a generalization of other proof systems (e.g., interactive proofs and probabilistically checkable proof) and for their concrete real-world application when used in conjunction with cryptographic tools.

Interactive oracle proofs (IOPs) are information-theoretic proof systems that combine aspects of interactive proofs (IPs) and probabilistically checkable proofs (PCPs). See [Gol08; Gol98] for more discussion on the definition and history of IPs and PCPs. Below we describe (public-coin) *IOPs of proximity* (IOPPs). We detail this general definition, and then use it to formally define its derivatives, such as IPs, PCPs and their proximity variants.

A k-round public-coin IOPP for a ternary relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ works as follows. The honest prover receives as input $(\mathbb{x}, \mathbb{y}, \mathbb{w})$, while the verifier receives as input $\mathbb{x}$ and oracle access to $\mathbb{y}$. In every round $i \in [\mathsf{k}]$, the verifier sends a uniformly random message α_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 $\mathbb{y}$ and proof strings $\Pi_1, \ldots, \Pi_{\mathsf{k}}$ sent by the prover, and then outputs a decision bit.

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

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

$$\Pr_{\alpha_1, \ldots, \alpha_{\mathsf{k}}} \left[\mathbf{V}^{\mathbb{y}, \Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{r} \Pi_1 \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{y}, \mathbb{w}) \\ \vdots \\ \Pi_{\mathsf{k}} \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \alpha_1, \ldots, \alpha_{\mathsf{k}}) \end{array} \right] = 1 \,.$$

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

$$\Pr_{\alpha_1,\ldots,\alpha_k}\left[\mathbf{V}^{\mathbb{y},\Pi_1,\ldots,\Pi_k}(\mathbb{x},\alpha_1,\ldots,\alpha_k)=1 \;\middle|\; \begin{array}{r}\Pi_1 \leftarrow \tilde{\mathbf{P}}(\alpha_1)\\ \vdots \\ \Pi_k \leftarrow \tilde{\mathbf{P}}(\alpha_1,\ldots,\alpha_k)\end{array}\right] \leq \beta(\mathbb{x},\mathbb{y})\,.$$

  When the soundness error depends only on the lengths of the inputs and on the proximity δ of $\mathbb{y}$ from the language $L_{\mathbb{x}} := \{\mathbb{y}' : \exists\, \mathbb{w},\ (\mathbb{x},\mathbb{y}',\mathbb{w}) \in \mathcal{R}\}$, we write $\beta(|\mathbb{x}|,|\mathbb{y}|,\delta)$ (and sometimes leave out $|\mathbb{x}|$ and $|\mathbb{y}|$, writing $\beta(\delta)$, when the lengths are clear from context).

We sometimes denote with bold letters a combination of proofs. For example, we let $\mathbf{\Pi} = (\Pi_1,\ldots,\Pi_k)$ denote the set of oracles received by the verifier. Given a set of queries Q to these oracles, $\mathbf{\Pi}[Q]$ is the set of symbols written in the appropriate oracles.

Efficiency measures. As IOPPs have many moving parts, they have many efficiency measures to keep track of. All of the complexity complexity measures described below 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.
- *Proof length* l: the total number of symbols in the proofs $\Pi_1,\ldots,\Pi_k$.
- *Input queries* $q_{\mathbb{y}}$: the number of alphabet elements read by the verifier from $\mathbb{y}$.
- *Proof queries* q_Π: the number of alphabet elements read by the verifier from $\Pi_1,\ldots,\Pi_k$.
- *Randomness* r: the verifier's i-th message α_i has length r_i and $r := \sum_{i=1}^{k} r_i$ 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.
- *Prover time* pt: $\mathbf{P}$ runs in time pt 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.

Derriving other proof systems from IOPPs. Given the formal definition of IOPs of proximity, we can derive the definitions of IPs and PCPs of proximity, and the standard decision variants of IOPs, IPs, and PCPs:

- *Probabilistically checkable proofs of proximity (PCPPs).* A PCPP is an IOPP where the prover sends a single message and then the verifier probabilistically reads it. It is common to refer to PCPPs as "non-interactive", since there is no communication beyond the single prover message. Note that PCPPs are distinct from 1-round IOPPs as defined above, since the prover goes first, whereas in the definition of IOPP we have set it so that the verifier speaks first.

- *Interactive proofs of proximity (IPPs).* An IPP is an IOPP where the verifier reads every symbol of the proofs sent to it by the prover. Formally, $l_{IOP} = \text{polylog}(|\mathbb{x}|)$ and $q_\Pi = l_{IOP}$.

- *Round-query IOPP.* A round-query IOPP is a hybrid proof system between an IOPP and an IPP. In a round-query IOPP, the verifier may decide which rounds it queries (as in an IOPP), but it must read these rounds in its entirety (as in an IPP). We let $\mathsf{q}_{\mathsf{round}}$ denote the "round-query" complexity of a round-query IOPP.
- *IOPs, PCPs, IPs, and round-query IOPs.* All of the proof systems defined above are the "proximity" variants of proof systems. We will sometimes be concerned with the "decision" variants of these proof systems. In the decision variant, the implicit input $\mathbb{y}$ is the empty string. Formally, they are proofs for relations of the form $\mathcal{R} = \{(\mathbb{x}, \perp, \mathbb{w})\}$. When this is the case, we generally omit $\perp$, which results in $\mathcal{R}$ being a binary relation. Furthermore $\mathsf{q}_{\mathbb{y}} = 0$ (as there is no implicit input to query), and so we omit the subscript in q_{Π} (as it is the only remaining query complexity) denoting simply $\mathsf{q} = \mathsf{q}_{\Pi}$. When we want to specify that we are referring to the decision variant of a proof system, we detract the word "proximity" from the name (e.g., IOP, PCP, etc.).

Proof systems for nondeterministic computation. Throughout our work (specifically, in Chapter 3), we will need to consider proof systems for non-deterministic computations. These are simply proof systems for the ternary relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w}) = ((M, \mathbb{x}, T), \mathbb{y}, \perp)\}$, where M is a Turing machine that receives two inputs $\mathbb{x}$ and $\mathbb{y}$, and T is a time bound. We have $((M, \mathbb{x}, T), \mathbb{y}, \perp) \in \mathcal{R}$ if $M(\mathbb{x}, \mathbb{y}) = 1$ in T time steps. We use the following theorem, showing that there exist very good PCPPs for non-deterministic computations:

Theorem 2.1.1 ([Mie09])**.** *There exists a non-adaptive PCPP for nondeterministic computations such that:*

PCPP for $M(\mathbb{x}, \mathbb{w}) = 1$ in T time steps	
Proof length	$\tilde{O}(T)$
Alphabet size	2
Queries	$O(1)$
Randomness	$O(\log T)$
Proximity	$O(1)$
Soundness error	$O(1)$
Prover running time	$\mathrm{poly}(T)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \log T)$

2.1.1 Round-by-round soundness

In some cases the standard definition of soundness does not suffice, and we need to consider stronger notions. In this thesis, the main variant of soundness that we will consider is *round-by-round* soundness, first defined for showing provably secure instantiations of the Fiat–Shamir heuristic in the standard model in [CCHLRRW19]. This notion has been very influential, spawning and much research (e.g., [CCHLRRW19; HLR21; BGTZ23]).

At the heart of the definition of round-by-round soundness is the notion of a "state function". Intuitively, the state function is a predicate that says whether a malicious prover is "winning" or "losing" in its goal of falsely convincing the verifier. Unlike in regular soundness, where we only consider whether the prover as succeeded in this venture at the end of the protocol, the state function allows for arguing this during the interaction between the prover and verifier.

Formally, let $(\mathbf{P}, \mathbf{V})$ be an IOPP for a relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$. A **state function** for $(\mathbf{P}, \mathbf{V})$ is a (possibly inefficient) function State that receives as inputs $\mathbb{x}$, $\mathbb{y}$, and a partial transcript tr and outputs a bit, and has the following properties:

- *Empty transcript:* if $\mathsf{tr} = \emptyset$ is the empty transcript, then $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 1$ if and only $(\mathbb{x}, \mathbb{y}) \in L(\mathcal{R})$.
- *Prover moves:* if tr is a transcript where the prover is about to move, and $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$ then, for every prover message Π, $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\Pi) = 0$.
- *Full transcript:* if tr is a full transcript and $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$, then $\mathbf{V}$ rejects given this interaction transcript.

Now that we have the definition of a state function, we define what it means for an IOPP to have round-by-round soundness. Informally, a protocol is round-by-round sound if at every point interaction, no matter what the prover does, it is unlikely that the verifier chooses randomness which changes the state from 0 to 1. If the state is 0 after the interaction is done, then the verifier can recognize this and will reject. We additionally define a "knowledge" variant of round-by-round soundness, which intuitively says that the only way that if the prover succeeds in causing the verifier to flip the state from 0 to 1, this is because it knows a witness for the instance (and, in particular, the input is in the relation).

- **Round-by-round soundness.** A k-round IOPP $(\mathbf{P}, \mathbf{V})$ for a relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ has *round-by-round knowledge soundness* with errors $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ if for every partial transcript $\mathsf{tr} = (\Pi_1, \alpha_1, \ldots, \Pi_{i-1}, \alpha_{i-1}, \Pi_i)$ up to round i where the verifier is about to move, with $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$ it holds that

$$\Pr_{\alpha_i}\left[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i)\right] < \varepsilon_i(\mathbb{x}, \mathbb{y})\,.$$

  If $\varepsilon_i \leq \beta_{\text{rbr}}$ for every i then we simply say that the IOPP has round-by-round soundness error β_{rbr}.

- **Round-by-round knowledge soundness** A k-round IOPP $(\mathbf{P}, \mathbf{V})$ for a relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ has *round-by-round knowledge soundness* with errors $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ and extraction time et if the IOPP has a state function State and there exists a deterministic "extractor" $\mathbf{E}$ that runs in time at most et with the following property: for every $\mathbb{x}$, $\mathbb{y}$ and transcript $\mathsf{tr} = (\Pi_1, \alpha_1, \ldots, \Pi_{i-1}, \alpha_{i-1}, \Pi_i)$, if
  - $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$, and

– $\Pr_{\alpha_i}[\mathsf{State}(\mathbb{x},\mathbb{y},\mathsf{tr}||\alpha_i)=1 \mid \mathsf{State}(\mathbb{x},\mathbb{y},\mathsf{tr})=0] > \varepsilon_i(\mathbb{x},\mathbb{y})$,
then $((\mathbb{x},\mathbb{y}),\mathbf{E}(\mathbb{x},\mathbb{y},\mathsf{tr})) \in \mathcal{R}$.

As with standard soundness, in both definitions we write ε_i as a function of proximity when appropriate.

2.1.2 IOPPs with decision randomness

In Chapter 3 it will be useful for us to consider an IOPP as being split into two distinct phases: in the *interaction phase* the prover and verifier interact and send their messages, and in the *decision phase* the verifier samples "decision randomness" α_{dc} and uses it to decide what to read of the interaction protocol. In particular, the prover does not play a part in the decision phase (it can be thought of as an offline post-processing phase). This will be useful for us both for considering separate amplification for errors that arrive from the interaction, and from errors that arise from the decision phase, and for considering proof systems where the verifier does not read its entire intraction randomness, but rather uses its decision randomness to read a few symbols from the messages it sent.

Formally, when considering separately the interaction and decision phases, we alter the definition of an IOPP as follows:

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

$$\Pr_{\alpha_1,\ldots,\alpha_k,\alpha_{\mathsf{dc}}}\left[\mathbf{V}^{\mathbb{y},\Pi_1,\ldots,\Pi_k,\alpha_1,\ldots,\alpha_k}(\mathbb{x};\alpha_{\mathsf{dc}})=1 \;\middle|\; \begin{array}{c}\Pi_1 \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w},\alpha_1)\\ \vdots \\ \Pi_k \leftarrow \mathbf{P}(\mathbb{x},\mathbb{w},\alpha_1,\ldots,\alpha_k)\end{array}\right] = 1\,.$$

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

$$\Pr_{\alpha_1,\ldots,\alpha_k,\alpha_{\mathsf{dc}}}\left[\mathbf{V}^{\mathbb{y},\tilde{\Pi}_1,\ldots,\tilde{\Pi}_k,\alpha_1,\ldots,\alpha_k}(\mathbb{x};\alpha_{\mathsf{dc}})=1 \;\middle|\; \begin{array}{c}\tilde{\Pi}_1 \leftarrow \tilde{\mathbf{P}}(\alpha_1)\\ \vdots \\ \tilde{\Pi}_k \leftarrow \tilde{\mathbf{P}}(\alpha_1,\ldots,\alpha_k)\end{array}\right] \leq \beta(\mathbb{x},\mathbb{y})\,.$$

When considering IOPs with decision randomness, we have two complexity measures that do not exist in standard IOPPs:
- *interaction randomness length* $\mathsf{r}_{\mathsf{IOP},\mathsf{int}}$: the total number of bits in $\alpha_1,\ldots,\alpha_k$.
- *decision randomness length* $\mathsf{r}_{\mathsf{IOP},\mathsf{dc}}$: The number of bits in α_{dc}.

Furthermore, we allow the query complexity $\mathsf{q}_{\mathsf{IOP}}$ to be the number of bits read by the verifier from $\alpha_1,\Pi_1,\ldots,\alpha_k,\Pi_k$ (i.e., we count queries to the verifier messages, and not only the prover messages).

Non-adaptive verifiers. A public-coin IOPP is *non-adaptive* if the algorithm run by $\mathbf{V}_{\mathsf{IOP}}$ after the interactive phase can be written as two algorithms $\mathbf{V}^{\mathsf{query}}_{\mathsf{IOP}}$ and $\mathbf{V}^{\mathsf{dec}}_{\mathsf{IOP}}$ such that:

- $\mathbf{V}^{\text{query}}$: Given $\mathbb{x}$ and randomness α, outputs Q, the set of queries made to the oracles of $\mathbf{V}$ on the same instance and randomness.

- $\mathbf{V}^{\text{dec}}$: Given $\mathbb{x}$, α and a set A of query answers, outputs the decision that $\mathbf{V}$ makes given instance $\mathbb{x}$, randomness α and A as the set of answers to its queries.

- Efficiency: Running $\mathbf{V}^{\text{query}}$ and $\mathbf{V}^{\text{dec}}$ one after the other has identical running time to running $\mathbf{V}$ on the same instance and randomness.

That is, for every instance $\mathbb{x}$, randomness $\alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IOP}}}, \alpha_{\mathsf{dc}}$ and oracles $\tilde{\mathbf{\Pi}} = (\tilde{\Pi}_1, \ldots, \tilde{\Pi}_k)$:

$$\mathbf{V}^{\tilde{\mathbf{\Pi}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}) = \mathbf{V}^{\text{dec}}\left(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}, \tilde{\mathbf{\Pi}}\left[\mathbf{V}^{\text{query}}\left(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}\right)\right]\right) .$$

Round-by-round soundness for IOPPs with decision randomness. In tandem with the alteration of the definition of IOPs, we augment the definition of round-by-round soundness. We change both the definition of a state function, and that of round-by round soundness

- **State function.** A state function for $(\mathbf{P}, \mathbf{V})$ with decision error δ_{dc} is a (possibly inefficient) Boolean function that receives as input an instance $\mathbb{x}$ and a transcript tr and outputs a bit for which the following holds.

  - *Empty transcript:* if $\mathsf{tr} = \emptyset$ is the empty transcript then $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ if and only if $\mathbb{x} \notin L(\mathcal{R})$.
  - *Prover moves:* if $\mathbb{x} \notin L(\mathcal{R})$, tr is a transcript where the prover is about to move and $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$, then for any potential prover message a, $\mathsf{state}(\mathbb{x}, \mathsf{tr}||a) = 0$.
  - *Full transcript:* if tr is a full transcript and $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ then $\Pr_{\alpha_{\mathsf{dc}}}[\mathbf{V}^{\mathsf{tr}}(\mathbb{x}; \alpha_{\mathsf{dc}}) = 1] \leq \delta_{\mathsf{dc}}$.

- **Round-by-round soundness.** An IOP $(\mathbf{P}, \alpha)$ with $\mathsf{k}_{\mathsf{IOP}}$ rounds for a relation $\mathcal{R}$ has $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}}, \delta_{\mathsf{dc}})$-round-by-round soundness if there exists a state function state with decision error δ_{dc} such that for all $\mathbb{x} \notin L(\mathcal{R})$, every $i \in [\mathsf{k}_{\mathsf{IOP}}]$, and every transcript tr of the first i rounds where the verifier is about to move and $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ it holds that

  $$\Pr_{\alpha}[\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha) = 1] \leq \varepsilon_i .$$

  We call $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ the interaction error and δ_{dc} the decision error. If $\delta_{\mathsf{dc}} = 0$, we omit δ_{dc} and simply say that IOP has round-by-round soundness $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$ which aligns with the standard definition of round-by-round soundness.

In Section 3.3, we show an amplification lemma for round-by-round soundness error for IOPPs with decision randomness similar to that which is known for IPs [CCHLRR18].

2.1.3 Polynomial IOPs

An important variant of IOPPs is *polynomial* IOPP. Informally, in a polynomial IOPP, the prover is restricted to sending as its messages low-degree polynomials. Knowing that the prover messages are structured in such a way tends to make it significantly easier to design proof systems with small soundness error. As such, polynomial IOPPs are usually used as part of a general paradigm where a polynomial IOPP is designed, and in order to compile it to a (standard) IOPP, we add additional proofs proving ensuring that the prover messages are (close to) representing low-degree polynomials. See Chapter 4 and Chapter 5 regarding the use of this paradigm.

Formally, a polynomial IOPP (poly-IOPP) is an IOPP $(\mathbf{P}, \mathbf{V})$ system where the prover (both honest and malicious) sends as its messages the evaluation of univariate polynomials over a finite field $\mathbb{F}$. In more detail, for every round i there is a prescribed list of ℓ_i degrees $(d_{i,j})_{j\in[\ell_i]})$ where $d_{i,j} \in \mathbb{N}$. In round i, the prover (both honest and malicious) outputs ℓ_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 $(\hat{f}_{i,j}(\mathbb{F}))_{j\in\ell_i}$ where $\hat{f}_{i,j}(\mathbb{F})$ is the evaluation of $\hat{f}_{i,j}$ over the entire field $\mathbb{F}$. In accordance with the change in the proof system, we have additional complexity parameters that will interest us:

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

When referring to the prover's messages we 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 simply denote that it has direct oracle access to $\hat{f}_{i,j}$.

Remark 2.1.2. When more convenient, we use polynomial IOPs where the degrees are bounded by the $d_{i,j}$ values (as opposed to being smaller than or equal to).

2.1.4 Interaction trees of probabilistic proofs

An alternate way of viewing a probabilistic proof is that of an interaction tree that describes all possible conversations between a prover and verifier. The (complete) ***interaction tree*** of an IOP $(\mathbf{P}, \mathbf{V})$ on an instance $\mathbb{x}$, denoted by $T_{\mathbb{x}}$, is defined as follows:

- The root represents the empty transcript.
- Each node v represents a transcript tr, and moving from node v to a child u via edge e corresponds to adding the message e to the transcript tr. Thus, the node u represents the transcript $\mathsf{tr}' = (\mathsf{tr}||e)$.
- Each leaf v, representing a complete transcript tr, is labelled with "accept" or "reject" matching whether $\mathbf{V}$ accepts tr given instance $\mathbb{x}$.

The value of an interaction tree $T_{\mathbb{x}}$, denoted by $\mathsf{val}(T_{\mathbb{x}})$, is the probability of reaching an accepting leaf from the root of the tree in a walk on the tree where verifier

messages are chosen uniformly at random and prover messages are chosen so as to maximize the probability of reaching an accepting node. The notion of value extends to sub-trees as well, where the value is the probability of reaching an accepting leaf when beginning on the root of the sub-tree. Notice that $\mathsf{val}(T_{\mathbb{x}}) = \max_{\tilde{\mathbf{P}}}\{\Pr[\langle\tilde{\mathbf{P}}, \mathbf{V}\rangle(\mathbb{x}) = 1]\}$.

Moreover, if $\mathbf{V}$ runs in polynomial time, then $\mathsf{val}(T_{\mathbb{x}})$ can be computed in space that is polynomial in $|\mathbb{x}|$ and in the round complexity, the proof length, and the verifier randomness of the IOP. This can be

2.1.5 IP to algorithm

Goldreich and Håstad [GH98] show that a public-coin IP with constant completeness and soundness errors for a relation implies a probabilistic algorithm that decides the same relation. Chiesa and Yogev [CY20] refine this result to apply to arbitrary completeness and soundness errors.

Lemma 2.1.3 ([GH98; CY20])**.** *Suppose that a relation $\mathcal{R}$ has a public-coin IP with completeness error α, soundness error β, round complexity k, and (binary) prover-to-verifier communication l. For $d(n) := \mathsf{l}(n) + \mathsf{k}(n) \cdot \log \frac{\mathsf{k}(n)}{1-\alpha(n)-\beta(n)}$, the relation $\mathcal{R}$ is in*

$$\mathrm{BPTIME}\left[2^{O(d)} \cdot \mathrm{poly}(n)\right].$$

2.2 Error correcting codes

A code of length n over an alphabet Σ is simply a subset $\mathcal{C} \subseteq \Sigma^n$. We generally think of codes in terms of taking a message Σ^k and encoding it into a codeword by adding redundancy. In this view, a code has error-correction properties if it can be decoded back into the correct message, even in the presence of corruptions in the codeword. Error correcting codes are an important tool, with applications ranging from the theoretical, for example in their extensive use in the development of the PCP theorem, to the practical. Indeed, virtually all information handled by modern computers is error corrected in one form or another. See [GRS23] for an in-depth exploration of coding theory.

More formally, an error correcting code consists of an tuple of (possibly inefficient) algorithms $(\mathbf{Enc}, \mathbf{Dec})$ where $\mathbf{Enc}$ is the encoding procedure, which takes a message in Σ^k and outputs a *codeword* in $\Sigma^{N(k)}$, and $\mathbf{Dec}$ is the decoding procedure, which takes a string in $\Sigma^{N(k)}$ (which we think of as a possibly corrupted codeword) and outputs a message in Σ^k (or, potentially, $\perp$ if the algorithm fails). The value $N(k)$ is known as the *block-length*. An error correcting code is called *linear* if $\Sigma = \mathbb{F}$ is a finite field, and $\mathcal{C} := \{\mathbf{Enc}(m) \mid m \in \Sigma^k\}$ is a subspace of $\mathbb{F}^{N(k)}$.

We say that the code is a $(N, \delta_{\mathsf{ECC}})$-code if it has block-length N and for every m and c' that is at Hamming distance at most δ_{ECC} from m it holds that $\mathbf{Dec}(c') = m$.

δ_{ECC} is the *unique-decoding distance* of the code, while $\rho = \frac{k}{N(k)}$ the *rate* of the code. Intuitively, codes with smaller rate (i.e., more redundancy) will have larger distance, and so will be preferred in some scenarios.

For $m = (m_1, \ldots, m_\ell) \in (\{0,1\}^k)^\ell$ we denote $\mathbf{Enc}(m, \ell) = \mathbf{Enc}(m_1), \ldots, \mathbf{Enc}(m_\ell)$ and similarly for $c = (c_1, \ldots, c_\ell) \in (\{0,1\}^{r(k)})^\ell$ with, $\mathbf{Dec}(c, \ell) = \mathbf{Dec}(c_1), \ldots, \mathbf{Dec}(c_\ell)$. The following theorem shows that there exist binary error-correcting codes with constant rate and distance:

Theorem 2.2.1 ([GI05]). *There exists a $(r, \delta_{\mathsf{ECC}})$-code over alphabet $\{0,1\}$ where $r(k) = O(k)$ and $\delta_{\mathsf{ECC}} = \Omega(1)$. Encoding and decoding k-bit strings takes time $\tilde{O}(k)$.*

When considering the decoding of a corrupted codeword, it may be that there are multiple codewords within distance radius δ. In this case, we would like to be able to argue that there are not many such codewords, and possibly output this (hopefully small) list. This is where we consider the notion of list-decodability. An error-correcting code $\mathcal{C}$ is (δ, ℓ)**-list decodable** if for every f the number of codewords in $\mathcal{C}$ within relative Hamming distance δ is at most ℓ.

The following simple 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 2.2.2. *If a code $\mathcal{C} \subseteq \mathbb{F}^n$ is (δ, ℓ)–list decodable, then any $\mathcal{C}' \subseteq \mathcal{C}$ is (δ, ℓ)–list decodable.*

2.2.1 Reed–Solomon and Reed–Muller codes

In this thesis, one family of error correcting codes will be of special importance, namely codes defined by the evaluation of low-degree polynomials. These codes are known as the Reed–Solomon and Reed–Muller codes, and were initially defined by their eponymous authors in [RS60] and [Mul54; Ree54]. Since their introduction in the early days of computer science, these codes have attracted significant attention and been important to countless research papers and real-world applications. See [GRS23] for extensive discussion on the Reed–Solomon and Reed–Muller codes.

In this thesis, most of our interest in these codes comes from the paradigm described in Section 2.1.3: we will design IOPPs for general computations in the polynomial IOPP model. When compiling the poly-IOPP into a standard IOPP, we will require that the prover send a Reed–Solomon (or Reed–Muller) encoding, rather than a polynomial, and, utilizing the amazing properties of polynomial-based error-correcting codes, will be able to test that the prover indeed sent such an encoding.

Definition 2.2.3. *The* **Reed–Solomon code** *over field $\mathbb{F}$, evaluation domain $\mathcal{L} \subseteq \mathbb{F}$, and degree $d \in \mathbb{N}$ is the set of evaluations over $\mathcal{L}$ of univariate polynomials (over $\mathbb{F}$) of degree less than d:*

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

The rate of $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is $\rho := d/|\mathcal{L}|$.

It is sometimes (e.g., in Chapter 4) convenient to talk about the Reed–Solomon code where the polynomials have degree *at most* d (rather than bounded by d). To differentiate these cases, we denote $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d] := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d-1]$ (i.e., RS' is the Reed–Solomon code where the degree is at most d.) Given a code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ and function $f\colon \mathcal{L} \to \mathbb{F}$, we sometimes use $\hat{f} \in \mathbb{F}^{<d}[X]$ to denote a nearest polynomial to f on $\mathcal{L}$ (breaking ties arbitrarily).

The Reed–Muller code can be seen as a generalization of Reed–Solomon codes to multivariate polynomials. While many definitions and notations are possible for Reed–Muller codes, in this thesis we will only be interested in bivariate Reed–Muller codes, and in specifying separately the individual degree of each variable:

Definition 2.2.4. *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:*

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

List-decoding Reed–Solomon codes. As in this thesis, the Reed–Solomon code will be our main code of interest, we will need to often consider its list-decoding. In order to streamline this, we use the following specialized notation: for a Reed–Solomon code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$, parameter $\delta \in [0,1]$, and $f \in \mathbb{F}^{\mathcal{L}}$, $\Lambda(f, d, \delta)$ denotes the list of codewords in $\mathcal{C}$ within relative Hamming distance at most δ from f. An important fact on the size of this list comes from the Johnson bound, which bounds the list size when decoding up to distance $1 - \sqrt{\rho}$:

Theorem 2.2.5 ([Joh62], see also [GRS23])**.** *The Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is $(1 - \sqrt{\rho} - \eta, \frac{1}{2\eta\sqrt{\rho}})$-list decodable for every $\eta \in (0, 1 - \sqrt{\rho})$, where $\rho := d/|\mathcal{L}|$ is the rate of the code.*

IOPs of proximity for constant-degree Reed–Solomon codes. We will utilize the existence of standard IOPPs for testing proximity to the Reed–Solomon code when the degree is constant. The following claim can be derived by the simple IOPP of the provers sending the verifier $d = O(1)$ field elements representing a degree d polynomial, and the verifier interpolating them, and comparing the evaluation of this polynomial with a random location in the function claimed to be a Reed–Solomon codeword:

Claim 2.2.6. *Let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ 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	$O(1)$

2.3 Randomness extractors

Randomness extractors allow for "extracting" (nearly) uniform randomness from non-uniform source distributions. While many notions exist, in this thesis we will consider source distributions with high min-entropy.

Definition 2.3.1. *The* **min-entropy** *of a random variable X is*

$$H_{\mathsf{min}}(X) = \min_{x \in \mathsf{supp}(X)} -\log \Pr[X = x]$$

We consider "two-source" extractors, the main source an n-bit string coming from a high-entropy, and the secondary source, the *seed*, is a d-bit string sampled according to the uniform distribution. We usually think of n as being much longer than d. Intuitively, the main source is where we would like to extract randomness from, and the seed can be used to "purify" the main source into one that is close to uniform.

Definition 2.3.2. *A function* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k,ε)**-extractor** *if for every random variable X with min-entropy at least k,* $\mathsf{SD}(\mathsf{Ext}(X, U_d), U_m) \leq \varepsilon$ *(where* SD *is the statistical distance). An extractor is explicit if it is computable in polynomial time.*

We use the following explicit construction of extractors with tight parameters.

Theorem 2.3.3 ([GUV09])**.** *For every constant $\alpha > 0$, and all positive integers n, k and all $\varepsilon > 0$, there is an explicit construction of a (k,ε)-extractor* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $d = O(\log n + \log(1/\varepsilon))$, and $m \geq (1-\alpha)k$.*

Setting specific parameters, we will use this simpler version of the theorem.

Theorem 2.3.4. *For all positive integers m, and $\ell \geq \log m$ there is an explicit construction of a $(2m, 2^{-\ell})$-extractor* $\mathsf{Ext}\colon \{0,1\}^{3m} \times \{0,1\}^d \to \{0,1\}^m$ *with $d = O(\ell)$.*

The following fact says that the number of strings within a Hamming-ball of radius γ around a string x is bounded, and will be useful in proofs in Chapter 3 that utilize randomness extractors.

Fact 2.3.5. *For all $n \in \mathbb{N}$, all $x \in \{0,1\}^n$ and $0 < \gamma < 1$ we have that*

$$\lvert\{x' \in \{0,1\}^n : \Delta(x,x') \leq \gamma)\}\rvert \leq 2^{n \cdot H(\gamma)}\,.$$

(here H is the entropy function $H(p) = -p\log(p) - (1-p)\log(1-p)$).

2.4 Exponential-time hypotheses

The exponential time hypothesis, introduced in [IP01], is a strengthening of the P $\neq$ NP conjecture roughly stating that 3SAT on n variables using deterministic algorithms requires exponential time to decide in the worst case. The exponential time hypothesis has proven to be a useful conjecture for proving barriers to improving algorithms, from faster algorithms for NP-complete problems like clique and vertex cover [IPZ01], to problems in fine-grained complexity (e.g., [Wil05]). In Chapter 6, using this family of conjectures in a similar manner, we will derive barriers on the efficiency of PCPs and IOPs.

Conjecture 2.4.1 ([IP01]). *The* **exponential time hypothesis (ETH)** *states that there exists a constant $c > 0$ such that* 3SAT $\notin$ DTIME$[2^{c \cdot n}]$.

As our main objects of study are probabilistic, it will be natural to consider a strengthening of the exponential time hypothesis which says that deciding 3SAT requires exponential time even for probabilistic algorithms:

Conjecture 2.4.2 ([CIKP08]). *The* **randomized exponential time hypothesis (RETH)** *states that there exists a constant $c > 0$ such that* 3SAT $\notin$ BPTIME$[2^{c \cdot n}]$.

2.5 Constraint satisfaction problems

Constraint satisfaction problems are inherently and deeply connected with probabilistic proof systems, most notably with PCPs. In Chapter 6 we will utilize this connection to show limitations of PCPs and IOPs. We denote by Σ a finite alphabet, l the number of variables, m the number of constraints, and q the constraint arity.

Definition 2.5.1. *A $(\Sigma, \mathsf{q}, \mathsf{l})$-**constraint** C is a tuple of indices $(i_1, \ldots, i_{\mathsf{q}})$ in $[\mathsf{l}]$ and a function $f \colon \Sigma^{\mathsf{q}} \to \{0,1\}$. An assignment $a \colon [\mathsf{l}] \to \Sigma$ **satisfies** the constraint C if*

$$f\big(a(i_1), \ldots, a(i_{\mathsf{q}})\big) = 1\,.$$

A constraint satisfaction problem is a list of constraints:

Definition 2.5.2. *A $(\Sigma, \mathsf{q}, \mathsf{l}, m)$-**CSP** is a list $\psi = (C_1, \ldots, C_m)$ where each C_i is a $(\Sigma, \mathsf{l}, \mathsf{q})$-constraint. We say that ψ is a (Σ, q)-CSP if it is a $(\Sigma, \mathsf{q}, \mathsf{l}, m)$-CSP for some l and m.*

The CSP ψ is **satisfiable** *if there exists an assignment that satisfies each of its constraints. The* **value** *of the CSP ψ is the maximum fraction of satisfied constraints across any assignment.*

It is a well-known fact that (non-adaptive) PCPs can be translated into CSPs. In this thesis we use an extension of this fact that works for PCPs where the verifier receives non-uniform advice. This extension follows straightforwardly from the standard PCP-to-CSP transformation.

Fact 2.5.3 (PCP to CSP). *Let $\mathcal{R}$ be a relation with a non-adaptive PCP with alphabet Σ, proof length l, query complexity q, randomness complexity r, decision time d, completeness error α, and soundness error β where the verifier uses ℓ bits of non-uniform advice.*

There exists a deterministic reduction that, given ℓ bits of non-uniform advice, receives as input an instance $\mathbb{x}$ for $\mathcal{R}$, runs in time $\mathrm{poly}(2^{\mathsf{r}}, \mathsf{d})$, and outputs a $(\Sigma, \mathsf{l}, \mathsf{q}, 2^{\mathsf{r}})$-CSP such that:
- *if $\mathbb{x} \in L(\mathcal{R})$ then the value of ψ is at least $1 - \alpha$;*
- *if $\mathbb{x} \notin L(\mathcal{R})$ then the value of ψ is at most β.*

Moreover, the size of each constraint in ψ is $\mathrm{poly}(\mathsf{d})$, and so $|\psi| = \mathrm{poly}(2^{\mathsf{r}}, \mathsf{d})$.

In Chapter 6 we utilize algorithms for efficiently solving CPSs with arity 2 and 3. Looking forward, we will use this to derive limitations to probabilistic proof systems: we will assume the existence of an extremely efficient IOP or PCP, reduce it into a CPS with small arity, and solve the CSP in polynomial time. If the IOP (resp. PCP) is efficient enough, this will imply a refutation to one of the exponential-time hypotheses.

Theorem 2.5.4 ([Sch78]). *There exists an algorithm that decides in time $\mathrm{poly}(|\psi|)$ whether a $(\{0,1\}, 2)$-CSP instance ψ is satisfiable.*

Theorem 2.5.5 ([Zwi98]). *There exists an algorithm that decides in time $\mathrm{poly}(|\psi|)$ whether a $(\{0,1\}, 3)$-CSP instance ψ has value 1 or value at most $5/8$.*

2.6 Pseudorandom generators

Pseudorandom generators allow for generating long pseudorandom strings out of short uniform randomness. In this thesis, we will be interested in generators whose output appears random to oracle-aided circuits. In particular, in Chapter 6 we will use this kind of pseudorandom generator to "fool" malicious IOP provers in order to derandomize IOPs as part of our toolbox of IOP-to-IOP transformations.

Definition 2.6.1. *For a boolean function f, an **f-oracle circuit** is a boolean circuit that includes f gates (in addition to standard gates). The size of the circuit is the total number of wires and gates.*

Definition 2.6.2. *A function $\mathsf{G}\colon \{0,1\}^{\ell_{\mathsf{PRG}}(n)} \to \{0,1\}^n$ is a **pseudorandom generator (PRG)** with seed length ℓ_{PRG}, computation time $\mathsf{t}_{\mathsf{PRG}}$, and advantage ϵ_{PRG} against f-circuits of size $\mathsf{s}_{\mathsf{PRG}}(n)$ if (i) G is computable by a Turing machine in time $\mathsf{t}_{\mathsf{PRG}}$; and (ii) for every f-circuit C of size at most $\mathsf{s}_{\mathsf{PRG}}(n)$ it holds that:*

$$|\Pr[C(U_n) = 1] - \Pr[C(\mathsf{G}(U_{\ell_{\mathsf{PRG}}})) = 1]| \leq \epsilon_{\mathsf{PRG}}\,.$$

Above U_n denotes the uniform distribution over binary strings of length n.

Results in [NW94; IW97] imply that, under certain complexity assumptions, PRGs with logarithmic seed length exist. It was later observed in [AIKS16] that these results can be extended to more powerful circuits.

Theorem 2.6.3 ([AIKS16]). *Let f be a boolean function. Assume that there exists a function in $\mathsf{E} = \mathrm{DTIME}(2^{O(n)})$ with f-circuit complexity $2^{\Omega(n)}$. Then, for every polynomial $\mathsf{s}_{\mathsf{PRG}}$, there exists a PRG $\mathsf{G}\colon \{0,1\}^{\ell_{\mathsf{PRG}}(n)} \to \{0,1\}^n$ against f-circuits of size $\mathsf{s}_{\mathsf{PRG}}(n)$, where $\ell_{\mathsf{PRG}}(n) = O(\log n)$ and $\epsilon_{\mathsf{PRG}} = 1/\mathsf{s}_{\mathsf{PRG}}(n)$.*

The assumption underlying the above theorem is a worst-case assumption that can be seen as a natural generalization of the assumption that $\mathsf{E} \not\subseteq \mathrm{NP}$. For a further discussion about this type of assumptions see [AIKS16; AASY16].

2.7 Additional useful facts

2.7.1 Polynomial identity lemma

We extensively use a simple form of the polynomial identity lemma, whose various variants are credited to (at the very least) Schwartz [Sch80], Zippel [Zip79], and DeMillo and Lipton [DL78].

Lemma 2.7.1. *For any non-zero polynomial $\hat{p} \in \mathbb{F}^{\leq d}[X]$ it holds that*

$$\Pr_{a \leftarrow \mathbb{F}}[\ \hat{p}(a) = 0\ ] \leq d/|\mathbb{F}|\,.$$

2.7.2 McDiarmid's inequality

Theorem 2.7.2 ([McD89]). *Let $X_1, X_2, \ldots, X_n$ be independent random variables such that $X_i \in \mathcal{K}_i$, for some measurable set $\mathcal{K}_i$. Suppose $f : \prod_{i=1}^n \mathcal{K}_i \to \mathbb{R}$ is 'Lipschitz' in the following sense: for each $k \leq n$ and any two input sequence $x, x' \in \prod_i \mathcal{K}_i$, that differ only in the k-th coordinate,*

$$|f(x) - f(x')| \leq \sigma_k.$$

Let $Y = f(X_1, X_2, \ldots, X_n)$. Then for every $\gamma > 0$,

$$\Pr[|Y - \mathbb{E}[Y]| \geq \gamma] \leq 2 \cdot \exp\left(-\frac{2\gamma^2}{\sum_{i=1}^n \sigma_i^2}\right).$$

2.7.3 A bound on division of binomials

Fact 2.7.3. *Let $n, k, q \in \mathbb{N}$ be parameters such that $q \leq k \leq n$. Then:*

$$\binom{n-q}{k-q} \Big/ \binom{n}{k} > (k/(e \cdot n))^q\,.$$

Proof.

$$\begin{aligned}
\binom{n-q}{k-q} \Big/ \binom{n}{k} &= \frac{(n-q)!}{(k-q)! \cdot (n-k)!} \cdot \frac{k! \cdot (n-k)!}{n!} \\
&= \frac{(n-q)!}{n!} \cdot \frac{k!}{(k-q)!} \\
&= \binom{k}{q} \Big/ \binom{n}{q} \\
&> \left(\frac{k}{q}\right)^q \cdot \left(\frac{q}{e \cdot n}\right)^q \\
&= (k/(e \cdot n))^q .
\end{aligned}$$

□

Chapter 3

How to be convinced while barely listening (even to yourself)

3.1 Introduction

Most research regarding IOPs has focused on understanding IOPs for languages in NP (and more generally various forms of non-deterministic computations) while using the additional rounds of interaction to achieve better efficiency compared to PCPs for those languages.

However, the power of IOPs for languages beyond NP is not well understood. We do know that IPs can express all languages in PSPACE for sufficiently large round complexity [LFKN92; Sha92]; moreover more rounds lead to more languages because, under plausible complexity assumptions, it holds that $\mathrm{IP}[\mathsf{k}] \not\subseteq \mathrm{IP}[o(\mathsf{k})]$ (while restricting to polynomial communication complexity) [GVW02]. But what can we say about the power of IOPs *with small query complexity (over the binary alphabet)*?[1] Not much is known about the power of general k-round IOPs, which leads us to ask:

What languages have a k*-round IOP with small bit-query complexity?*

We answer the above question by showing that (informally) the power of IOPs with k rounds where the verifier reads $O(1)$ bits from $O(\mathsf{k}/\log|\mathbb{x}|)$ of the communication rounds (both prover and verifier messages) is the same as if the verifier reads the entire protocol transcript (as in an IP). This can be seen as extending the PCP Theorem to interactive proofs, interpreted as "*you can be convinced by a conversation while barely listening (even to yourself)*".

Theorem 1. *Let L be a language with a k-round public-coin IP. Then L has a k-round public-coin binary IOP where, on input $\mathbb{x}$, the verifier reads $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$ bits of the interaction transcript. All other parameters are polynomially related.*

[1] An IP is an IOP where the verifier has large query complexity over the binary alphabet.

Theorem 1 is surprising in light of the work of Goldreich, Vadhan, and Wigderson [GVW02]. Since the query complexity is smaller than the round complexity, Theorem 1 implies that the IOP verifier does not make queries to every round of the protocol. This can be viewed as saying that the power of AM[k] is unchanged even when the verifier only accesses $O(\mathsf{k}/\log|\mathbb{x}|)$ of the k rounds. In other words, reducing round complexity of public-coin protocols reduces their power, but it is nevertheless possible to not read every round of interaction while preserving the power.

IOPs from interactive reducibility. Theorem 1 establishes that AM$[O(\log|\mathbb{x}|)]$ has IOPs with $O(\log|\mathbb{x}|)$ rounds and constant query complexity (over the binary alphabet). However, it does not provide IOPs with constant query complexity from IPs with $\omega(\mathsf{k})$ rounds.

We extend this by showing IOPs with constant query complexity for languages that are *interactively reducible*, a notion that we introduce in this work. Informally, the notion requires that it is possible to reduce, via an interactive protocol, multiple transcripts of an IP for the relation into a single transcript. We require that the probability of the verifier accepting conditioned on the reduced transcript be (roughly) the minimum probability of the verifier accepting conditioned on any of the original transcripts. See Section 3.2.2 for further details and discussion.

The notion of interactive reducibility allows us to get an optimal version of Theorem 1 for additional languages. Let IR[k] be the class of languages that have a (1-round) interactive reduction with k predicates (these predicates roughly align with rounds of an IP).

Theorem 2 (informal)**.** *Let L be a language in* IR[k]*. Then L has an $O(\mathsf{k})$-round public-coin IOP, with polynomial proof length, where the verifier reads $O(1)$ bits of the interaction transcript.*

Interactive reducibility is a natural property; we show that general interactive proofs can be seen as interactive reductions (albeit ones with bad parameters), and show efficient interactive reductions for the sumcheck protocol [LFKN92] and for Shamir's protocol [Sha92].

Thus, applying Theorem 2 to the sumcheck protocol yields the following (perhaps surprising) conclusion: *any k-round sumcheck protocol can be transformed to a $O(\mathsf{k})$-round IOP where the verifier has $O(1)$ query complexity (over the binary alphabet).* Notice that using standard PCP techniques it is only known how to achieve a similar (non-interactive) result with *exponential* proof length.

Similarly, using our interactive reduction for Shamir's protocol, we recover the result of [CFLS97], showing a constant query poly-round IOP for PSPACE.

Related work. The PCP Theorem can be viewed as a "half-round" IOP with query complexity $O(1)$ and decision randomness $O(\log n)$ for NP. For languages above NP, prior works imply certain facts about k-round IOPs for extreme settings of k.

- For languages that have a public-coin IP with $\mathsf{k} = 1$ round (a verifier message followed by a prover message), Drucker [Dru11a] proves a hardness of approximation result in the terminology of CSPs. His result can be re-interpreted showing

that these languages have a one-round IOP where the verifier reads $O(1)$ bits from each message and decides (using $O(\log n)$ bits of decision randomness). However, Drucker's result does not extend to arbitrary many rounds.[2]

- When k can be polynomially large, we observe that constant-query IOPs for PSPACE can be obtained from [CFLS95; CFLS97],[3] which in turn provides such an IOP for every language having an IP. Other analogues of PCP have been given (e.g., [HRT07] applies to the polynomial hierarchy, [Dru11b] is also for PSPACE) but they do not seem to translate to IOPs.
- For general k, one can use the fact that $\mathrm{AM}[k] \subseteq \mathrm{NEXP}$, and obtain a PCP where the prover sends a single *exponentially-long* message from which the (polynomial-time) verifier reads $O(1)$ bits. However, this does not help if we require the prover to send messages of *polynomial length*.

See Figure 3.1 for a table summarizing these results and ours.

	complexity class/language	**proof length**	**query complexity**	**round complexity**
[BGHSV05]	NEXP	$\exp(\lvert x\rvert)$	$O(1)$	N/A (PCP)
[CFLS97], **[this work]**	PSPACE	$\mathrm{poly}(\lvert x\rvert)$	$O(1)$	$\mathrm{poly}(\lvert x\rvert)$
[this work]	$\#\mathrm{SAT}_k$	$\mathrm{poly}(\lvert x\rvert)$	$O(1)$	$O(k)$
[BGHSV05] (implied)	AM[k]	$\exp(\lvert x\rvert)$	$O(1)$	N/A (PCP)
[this work]	AM[k]	$\mathrm{poly}(\lvert x\rvert)$	$\min\{O\left(\frac{k}{\log \lvert x\rvert}\right), O(1)\}$	$O(k)$
[Dru11a]	AM	$\mathrm{poly}(\lvert x\rvert)$	$O(1)$	1
[ALMSS98; AS98]	NP	$\mathrm{poly}(\lvert x\rvert)$	$O(1)$	N/A (PCP)

Figure 3.1: Classes captured by different types of probabilistic proofs (in the regime of constant soundness error). Here, x denotes the instance whose membership in the language the verifier is deciding. Here, AM stands for two-message public-coin protocols (a verifier random message followed by a prover message), and AM[k] is a k-round public-coin protocol.

3.2 Techniques

In this section, we describe our main transformation:

[2] Round reduction [BM88] can reduce the number of rounds from any k to 1 with a blow-up in communication that is exponential in k. This does not work when k is super constant; see Section 3.2.4.3 for further discussion.

[3] Their result shows that PSPACE has what is known as a *probabilistically checkable debate system*. In their system, one prover plays a uniform random strategy. Thus one can naturally translate the debate system into an IOP.

- In Section 3.2.1, we show how to transform k-round IPs into round-query IOPs with (round-)query complexity $O(k/\log|\mathbb{x}|)$ (these are IPs where the verifier can choose which rounds to read following the interaction).
- In Section 3.2.2 we define interactive reducibility and extend the previous results to any language that is interactively reducible.
- In Section 3.2.3 we discuss how to transform an AM[2] interactive proof into an IOP.
- In Section 3.2.4 we discuss how to allow the verifier to read few bits of its own randomness.
- In Section 3.2.5 we define index-decodable PCPs, which will help us get local access to prover messages.
- In Section 3.2.6 we show how to use index-decodable PCPs to acheive local access to prover messages.
- In Section 3.2.7 we show how to construct index-decodable PCPs.

3.2.1 IPs to round-query IOPs

We outline how to transform an IP into an IOP with small query complexity. We first show how to transform a logarithmic-round IP into a round-query IOP where the verifier queries $O(1)$ rounds.[4] Then we extend this idea to transform a k-round IP into a round-query IOP with $O(k/\log|\mathbb{x}|)$ round-query complexity.

3.2.1.1 From $O(\log|\mathbb{x}|)$-round IP to $O(1)$-query IOP

We show how to transform a k-round public-coin IP where $k = O(\log|\mathbb{x}|)$ into an $O(k)$-round public-coin IOP with the following efficiency: polynomial proof length over the binary alphabet; constant query complexity; and logarithmic decision randomness.

First, we sketch how to transform a k-round public-coin IP $(\mathbf{P}_{IP}, \mathbf{V}_{IP})$ into an $O(k)$-round public-coin IOP $(\mathbf{P}_{IOP}, \mathbf{V}_{IOP})$ where the verifier reads $O(1)$ rounds (in their entirety) from the interaction transcript. Then we explain how to ensure that the verifier queries $O(1)$ bits in total.

A strawman protocol. We describe a natural strategy for transforming the IP into an IOP where the verifier reads $O(1)$ rounds, albeit with high soundness error. The IOP prover $\mathbf{P}_{IOP}$ and IOP verifier $\mathbf{V}_{IOP}$ respectively simulate the IP prover and verifier $(\mathbf{P}_{IP}, \mathbf{V}_{IP})$, inducing an interaction transcript $tr = (a_1, a_1, \ldots, a_k, a_k)$. At this time, however, $\mathbf{V}_{IOP}$ *does not read any messages from* tr. After this interaction, $\mathbf{P}_{IOP}$ sends a transcript tr', which is allegedly equal to tr, as a single message. Then $\mathbf{V}_{IOP}$ reads tr' and checks that tr' is an accepting transcript for the IP verifier $\mathbf{V}_{IP}$. Moreover, $\mathbf{V}_{IOP}$ tests consistency between tr (the real interaction) and tr' (the alleged copy of the interaction sent as a single message): $\mathbf{V}_{IOP}$ samples a random $i \in [k]$, reads the i-th prover message and

[4] Recall that a round-query IOP is an IOP where the verifier can read fewer than k rounds, but any round that is queried is read in its entirety.

i-th verifier message in tr, and checks that these equal the corresponding messages in tr'.

In this IOP, $\mathbf{V}_{\mathsf{IOP}}$ reads $O(1)$ messages from the interaction transcript, but the soundness error of the IOP is large (even when discounting the soundness error of the underlying IP). Indeed, it may be that a cheating IOP prover sends a malicious transcript tr' that is accepting but differs from the real transcript tr in one round only. In this case, $\mathbf{V}_{\mathsf{IOP}}$ catches the inconsistency only with probability $1/\mathsf{k}$, which means that the soundness error could be as large as $1 - 1/\mathsf{k}$.

Note that reducing this soundness error via parallel repetition would increase the number of rounds queried by $\mathbf{V}_{\mathsf{IOP}}$. Achieving constant soundness error would require $O(\mathsf{k})$ repetitions, resulting in an IOP verifier that reads $O(\mathsf{k})$ rounds, taking us back to where we started.

Our transformation. We present a transformation that improves on the above strawman protocol, achieving a constant soundness error for any IP that has a logarithmic number of rounds.

A malicious IOP prover in the strawman protocol has two strategies: the transcript tr' sent as the last message either agrees with the real transcript tr on more than half of the rounds, or it does not. If tr' agrees with tr in less than half of the rounds, then the IOP verifier catches this inconsistency with probability at least $1/2$. Intuitively, the transformation that we sketch below ensures that if tr' is consistent with tr on at least half of the rounds, then the consistent rounds must contain within them a full execution of the underlying IP. Then, since this consistent part was generated interactively in tr, by the soundness property of the IP, this contained transcript will be rejected with high probability. We now describe our transformation in more detail.

Suppose that our public-coin IP has $\mathsf{k}(|\mathbb{x}|) = O(\log |\mathbb{x}|)$ rounds and that the verifier message in each round is r bits long. The IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ on a given instance $\mathbb{x}$ works as follows.

1. $\mathbf{P}_{\mathsf{IOP}}$ sets $S_0 := \{\emptyset\}$ (i.e., S_0 consists of the empty transcript).
2. For $i = 1, \ldots, 2\mathsf{k}$:
   - $\mathbf{P}_{\mathsf{IOP}}$ sends S_{i-1}.
   - $\mathbf{V}_{\mathsf{IOP}}$ sends a random $\alpha_i \in \{0,1\}^{\mathsf{r}}$ (corresponding to a message of $\mathbf{V}_{\mathsf{IP}}$).
   - $\mathbf{P}_{\mathsf{IOP}}$ sets $S_i := S_{i-1} \cup \{(\mathsf{tr}||\alpha_i||a_{\mathsf{tr},i})\}_{\mathsf{tr} \in S_{i-1}}$ where $a_{\mathsf{tr},i} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}||\alpha_i)$ for each $\mathsf{tr} \in S_{i-1}$.
3. $\mathbf{P}_{\mathsf{IOP}}$ sends $S_{2\mathsf{k}}$ and, for every $i \in [2\mathsf{k}]$, also sends $T_i := S_i$.
4. In the decision phase, $\mathbf{V}_{\mathsf{IOP}}$ performs the checks below.
   (a) *Subset consistency:* For every $i \in [2\mathsf{k}]$, check that $T_{i-1} \subseteq T_i$.
   (b) *Transcript consistency:* Choose a random $i \in [2\mathsf{k}]$. Check that $S_{i-1} = T_{i-1}$ and $S_i = T_i$. Additionally, check that for every $\mathsf{tr} \in S_{i-1}$ there is a message $a_{\mathsf{tr},i}$ such that $(\mathsf{tr}||\alpha_i||a_{\mathsf{tr},i}) \in S_i$, where α_i is the verifier message sent during the i-th round of interaction.
   (c) *Membership:* Check that for every transcript $\mathsf{tr} \in T_{2\mathsf{k}}$ that is complete (i.e.,

contains messages for all k rounds of the IP) it holds that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}) = 1$.

Efficiency. We briefly discuss the main efficiency measures of the transformation.

- *Query complexity.* The IOP verifier reads $O(1)$ rounds from the transcript.
- *Communication complexity.* We argue that all messages in the protocol have length $\text{poly}(|\mathbb{x}|)$. For every i, $|S_i| \leq 2|S_{i-1}|$ since S_i contains all transcripts in S_{i-1} and continuations of each of these transcripts. Since $\mathsf{k} = O(\log|\mathbb{x}|)$, $|S_i| = \text{poly}(|\mathbb{x}|)$ for every i. Each transcript within S_i has polynomial length, so the length of these messages is $\text{poly}(|\mathbb{x}|)$. Finally, the sets $T_1, \ldots, T_{2\mathsf{k}}$ have the same sizes as $S_1, \ldots, S_{2\mathsf{k}}$ (respectively) and so the prover's final message has length $\text{poly}(|\mathbb{x}|)$.
- *Decision randomness.* The IOP verifier uses $O(\log \mathsf{k}) = O(\log|\mathbb{x}|)$ bits of decision randomness.

Analysis. In this overview, we discuss soundness only, as completeness follows straightforwardly from the construction. Let β be the soundness error of $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$. We show that the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ has soundness error

$$\beta_{\mathsf{IOP}} = \max\left\{\frac{1}{2}, \binom{2\mathsf{k}}{\mathsf{k}} \cdot \beta\right\}.$$

The above expression can be made constant by applying $\text{poly}(|\mathbb{x}|)$ parallel repetitions to $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ prior to applying our transformations, until it has soundness error $\beta' \leq \left(2 \cdot \binom{2\mathsf{k}}{\mathsf{k}}\right)^{-1}$.

Fix $\mathbb{x} \notin L$ and a cheating IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. A fixed transcript of the IOP has the structure:

$$\left(S_0, \alpha_1, S_1, \ldots, \alpha_{2\mathsf{k}}, S_{2\mathsf{k}}, (T_0, \ldots, T_{2\mathsf{k}})\right).$$

Given such a transcript, we say that an index i is *consistent* if: (a) $S_{i-1} = T_{i-1}$ and $S_i = T_i$; and (b) for every $\mathsf{tr} \in S_{i-1}$ there is a message $a_{\mathsf{tr},i}$ such that $(\mathsf{tr}||\alpha_i||a_{\mathsf{tr},i}) \in S_i$.

Conditioned on the event that the transcript generated during the interaction has less than k consistent indices i, $\mathbf{V}_{\mathsf{IOP}}$ rejects with probability at least $1/2$ due to its check in Item 4b. We are thus left to analyze the probability that $\mathbf{V}_{\mathsf{IOP}}$ rejects conditioned on the event that the generated transcript has at least k consistent indices.

Fix indices $i_1 < \cdots < i_{\mathsf{k}}$ and suppose that all these indices are consistent with respect to the transcript. By the definition of consistency, for every $j \in [\mathsf{k}]$, $S_{i_j-1} = T_{i_j-1}$, $S_{i_j} = T_{i_j}$ and $(\mathsf{tr}||\alpha_{i_j}||a_{\mathsf{tr},i_j}) \in S_{i_j}$ for every $\mathsf{tr} \in S_{i_j-1}$. This implies that there exist $a_{i_1}, \ldots, a_{i_{\mathsf{k}}}$ such that $(\alpha_{i_1}||a_{i_1}||\cdots||\alpha_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) \in T_{i_{\mathsf{k}}}$. By the subset consistency check, $\mathbf{V}_{\mathsf{IOP}}$ accepts only if $T_{i_{\mathsf{k}}} \subseteq T_{2\mathsf{k}}$, in which case $(\alpha_{i_1}||a_{i_1}||\cdots||\alpha_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) \in T_{2\mathsf{k}}$. This transcript was generated interactively by the prover and verifier and hence, by the soundness of the IP, $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha_{i_1}||a_{i_1}||\cdots||\alpha_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) = 1$ with probability at most β. If the transcript is rejecting, then this is detected by $\mathbf{V}_{\mathsf{IOP}}$ in its membership check.

The previous analysis holds for fixed indices $i_1 < \cdots < i_k$. By applying the union bound to all choices of indices, we have that, conditioned on the transcript generated having at least k consistent rounds, $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\binom{2k}{k} \cdot \beta$.

Putting together both (non-intersecting) events of the number of rounds consistent with the generated transcript, we conclude that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\max\{\frac{1}{2}, \binom{2k}{k} \cdot \beta\}$.

Remark 3.2.1. As alluded to above, the above transformation can be viewed as two steps: (i) apply a transformation that ensures that (with constant probability) any transcript tr' that agrees with the real interaction transcript tr on at least half of the rounds is rejecting; and (ii) apply the strawman protocol to the new protocol.

We believe that this property of rejecting transcripts that are close to the real interaction transcript, which we call "robust soundness error", is of independent interest and is likely to have further applications. A formal definition of (round-)robust soundness follows.

Definition 3.2.2 ((Round-)Robust soundness)**.** *Let* $\mathrm{IP} = (\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ *be an IP for a relation* $\mathcal{R}$*.* IP *has* **(round-)robust soundness error** ε **with distance** δ *if for every instance* $\mathbb{x} \notin L(\mathcal{R})$ *and malicious prover* $\tilde{\mathbf{P}}_{\mathsf{IP}}$*:*

$$\Pr_{\alpha_1,\ldots,\alpha_k}\left[\begin{array}{l|r} \exists\, \mathsf{tr}' : \Delta_{\mathsf{Round}}(\mathsf{tr},\mathsf{tr}') \le \delta \ \wedge\ \mathbf{V}_{\mathsf{IP}}(\mathbb{x},\mathsf{tr}') = 1 & \begin{array}{r} a_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IP}}(\alpha_1) \\ \vdots \\ a_k \leftarrow \tilde{\mathbf{P}}_{\mathsf{IP}}(\alpha_1,\ldots,\alpha_k) \\ \mathsf{tr} := (\alpha_1,a_1,\ldots,\alpha_k,a_k) \end{array} \end{array}\right] \le \varepsilon\,,$$

where $\Delta_{\mathsf{Round}}(\mathsf{tr},\mathsf{tr}')$ *is the fraction of rounds on which* tr *and* tr' *differ.*

Other notions of robustness for IPs can be considered by using different distance measures between tr and tr' (e.g., Hamming distance, distance between groups of rounds, and so on).

3.2.1.2 Round-query IOPs from general IPs

The transformation described in the previous section works for $O(\log|\mathbb{x}|)$-round IPs. However, it cannot be directly applied to IPs with more rounds because the proof length (and thus also the verifier running time) would be more than polynomial.

Nevertheless, we extend the transformation to work for any public-coin IP while achieving a moderate improvement on the number of read rounds. In the main loop of the IOP, rather than advancing each IP transcript in S_{i-1} by one round, advance it by $O(k/\log|\mathbb{x}|)$ rounds before inserting the resulting transcripts into the set S_i. During its decision phase, the IOP verifier chooses an index i and reads the entire $O(k/\log|\mathbb{x}|)$-round interaction done during this iteration of the IOP. Completeness and soundness of this new transformation are similar to the one presented in the previous section, but now the verifier reads $O(k/\log|\mathbb{x}|)$ rounds. The rest of the

efficiency parameters are similar to the IOP described in the previous section, except that proof length is polynomial regardless of k.

Can we do better? The construction described in this section doubles the number of transcripts stored in the set S_i relative to S_{i-1}. This causes a blow-up in parameters and is the reason why this approach fails in constructing $O(1)$-query IOPs from IPs with super-logarithmic round complexity. Intuitively, if we could reduce this doubling then we may be able to modify the transformation to get $O(1)$-query IOPs. While we do not achieve this for *general* IP, we show that, using this intuition, we can construct $O(1)$-query IOPs for a rich class of relations that are *interactively reducible*. We discuss this notion, and corresponding new IOP constructions, in the following section.

3.2.2 Interactive reducibility

In Section 3.2.1.1 we described how to transform an IP into an IOP with $O(1)$ query complexity. The new protocol kept track of a set containing all partial transcripts of the IP generated so far in the protocol. In every round, every partial transcript in the set was advanced by one round, and the newly advanced transcripts were added to the set held previously. This meant that in every round, the set contains twice as many transcripts as in the previous round. As we require polynomial proof length and verifier running time, this technique was unable to allow reading of $O(1)$ rounds for IPs with greater than $O(\log |\mathbb{x}|)$ rounds.

Intuitively, suppose it were possible to take multiple transcripts and reduce them into a single transcript that in some sense preserves soundness of all of the transcripts combined. In that case, this issue could be bypassed, and the protocol would work for IPs with super-logarithmic round complexity. In more detail, suppose we want to reduce transcripts $\mathsf{tr}_1, \ldots, \mathsf{tr}_t$ into a new transcript tr'. The *acceptance probability* of a transcript prefix tr_i is the maximum over all prover strategies of the probability that the verifier will end up accepting when continuing interaction with the prover from transcript tr_i. Roughly, we require that: (a) if the acceptance probability of every tr_i is 1, then so is the acceptance probability of tr'; and (b) if there exists some transcript tr_i with small acceptance probability, then (with high probability) the acceptance probability of tr' is also small.

In this section, we introduce the concept of *interactive reducibility*, which formally captures this intuition. We exemplify this in Section 3.2.2.1 by describing how to reduce multiple transcripts of the sumcheck protocol into a single transcript. Then, in Section 3.2.2.2, we formally define interactive reducibility. In Section 3.2.2.3, we show how to adapt the protocol described in Section 3.2.1.1 to work with interactive reductions and bypass the blow-up in the original protocol. Finally, in Section 3.2.2.4, we discuss relations known to have interactive reducibility.

3.2.2.1 An interactive reduction for sumcheck

We exemplify the notion of interactive reducibility in the case of the sumcheck protocol [LFKN92]. Below we review this protocol, and then explain how to reduce multiple transcripts into one transcript via an interactive reduction.

The sumcheck protocol. The verifier has query access to a n-variate polynomial p of individual degree d over some field $\mathbb{F}$. The goal of the verifier is to test, for a given field element γ, whether

$$\sum_{\alpha_1,\ldots,\alpha_n \in \{0,1\}} p(\alpha_1,\ldots,\alpha_n) = \gamma\,.$$

The protocol begins with the prover sending a polynomial $\tilde{p}_1$ of degree d, claimed to equal $p_1(X) := \sum_{\alpha_2,\ldots,\alpha_n \in \{0,1\}} p(X,\alpha_2,\ldots,\alpha_n)$. The verifier checks that $\tilde{p}_1(0) + \tilde{p}_1(1) = \gamma$ (rejecting if not), samples a random field element r_1, and sends it to the prover. Both parties define $\gamma_1 := \tilde{p}_1(r_1)$.

This one-round interaction leads to a new sumcheck claim

$$\sum_{\alpha_2,\ldots,\alpha_n \in \{0,1\}} p(r_1,\alpha_2,\ldots,\alpha_n) = \gamma_1\,,$$

that has the following properties: (a) if the original claim is true then the new claim is also true; and (b) if the original claim is false then with high probability the new claim is also false.

Next, the prover sends $\tilde{p}_2$ claimed to equal $p_2(X) := \sum_{\alpha_3,\ldots,\alpha_n \in \{0,1\}} p(r_1,X,\alpha_3,\ldots,\alpha_n)$ and the protocol repeats as before. This process continues until the n variables are fixed to some field elements $(r_1,\ldots,r_n)$, and the problem has been reduced to checking that $p(r_1,\ldots,r_n) = \gamma_n$, which the verifier can check via one query to the polynomial p.

One can associate a round j of the protocol with a list of field elements $(r_1,\ldots,r_j)$ and a claimed sum γ_j, and think of that round as "reducing" a claim $\mathbb{z} = ((r_1,\ldots,r_j),\gamma_j)$ that

$$\sum_{\alpha_{j+1},\ldots,\alpha_n \in \{0,1\}} p(r_1,\ldots,r_j,\alpha_{j+1},\ldots,\alpha_n) = \gamma_j\,,$$

into a new claim $\mathbb{z}' = ((r_1,\ldots,r_{j+1}),\gamma_{j+1})$ that

$$\sum_{\alpha_{j+2},\ldots,\alpha_n \in \{0,1\}} p(r_1,\ldots,r_{j+1},\alpha_{j+2},\ldots,\alpha_n) = \gamma_{j+1}\,.$$

Reducing multiple sumcheck claims. We are given claims $\mathbb{z}_1,\ldots,\mathbb{z}_t$ where each $\mathbb{z}_i$ consists of $(r_{i,1},\ldots,r_{i,j})$ and a claimed sum $\gamma_{i,j}$. We seek to reduce these t claims into a single claim $\mathbb{z}' = ((r'_1,\ldots,r'_{j+1}),\gamma'_{j+1})$ such that: (a) If each $\mathbb{z}_i$ is a true statement, then $\mathbb{z}'$ is a true statement; and (b) If there is some $\mathbb{z}_i$ that is a false statement, then with high probability $\mathbb{z}'$ is a false statement. Notice that in order to merge multiple

sumcheck transcripts it suffices to merge multiple sumcheck claims $\mathbb{z}_1, \ldots, \mathbb{z}_t$. Thus we will focus on merging such claims.

Before describing the reduction, we define a polynomial $I_{\mathbb{z}_1,\ldots,\mathbb{z}_t} : \mathbb{F} \to \mathbb{F}^j$ that represents a curve through the t points described by the instances $\mathbb{z}_1, \ldots, \mathbb{z}_t$. That is, $I_{\mathbb{z}_1,\ldots,\mathbb{z}_t}(i) = (r_{i,1}, \ldots, r_{i,j})$ for every $i \in [t]$ (here we implicitly associate the set $[t]$ with an arbitrary set $S \subseteq \mathbb{F}$ of size t, known to all parties). By interpolation, the degree of $I_{\mathbb{z}_1,\ldots,\mathbb{z}_t}$ is less than t.

Given this definition, we describe the interactive reduction for the sumcheck protocol.

- Prover: Send the polynomial $g \in \mathbb{F}[X_1, X_2]$ defined as:

$$g(X_1, X_2) := \sum_{\alpha_{j+2},\ldots,\alpha_n \in \{0,1\}} p(I_{\mathbb{z}_1,\ldots,\mathbb{z}_t}(X_1), X_2, \alpha_{j+2}, \ldots, \alpha_n)\,. \tag{3.1}$$

- Verifier: Receive a bivariate polynomial $\tilde{g} \in \mathbb{F}[X_1, X_2]$ of degree at most $j \cdot d \cdot (t-1)$ in X_1 and degree at most d in X_2.
  1. *Consistency:* Check that for every $i \in [t]$ it holds that $\sum_{\alpha \in \{0,1\}} \tilde{g}(i, \alpha) = \gamma_{i,j}$. (Reject if not.)
  2. *Generate new instance:*
     (a) Sample uniformly random field elements $\alpha, r^* \leftarrow \mathbb{F}$ and send them to the prover.
     (b) Set $(r'_1, \ldots, r'_j) := I_{\mathbb{z}_1,\ldots,\mathbb{z}_t}(\alpha)$ and $\gamma_{j+1} := \tilde{g}(\alpha, r^*)$ and output the new instance

$$\mathbb{z}' := \Big((r'_1, \ldots, r'_j, r^*), \gamma_{j+1}\Big)\,.$$

Analysis. It follows straightforwardly from the protocol that, if $\mathbb{z}_1, \ldots, \mathbb{z}_t$ are all true statements and the prover acts honestly, then $\mathbb{z}'$ is a true statement. We show that if any one of the statements $\mathbb{z}_1, \ldots, \mathbb{z}_t$ is false then with high probability so is $\mathbb{z}'$.

Let g be as defined in Equation 3.1 with respect to $\mathbb{z}_1, \ldots, \mathbb{z}_t$. Suppose that $\mathbb{z}_i$ is a false claim (i.e., $\sum_{\alpha_{j+1},\ldots,\alpha_n \in \{0,1\}} p(r_{i,1}, \ldots, r_{i,j}, \alpha_{j+1}, \ldots, \alpha_n) \neq \gamma_{i,j}$). Then, by definition,

$$\sum_{\alpha \in \{0,1\}} g(i, \alpha) = \sum_{\alpha_{j+1},\ldots,\alpha_n \in \{0,1\}} p(r_{i,1}, \ldots, r_{i,j}, \alpha_{j+1}, \ldots, \alpha_n) \neq \gamma_{i,j}\,.$$

During its consistency check, the verifier checks that $\sum_{\alpha \in \{0,1\}} \tilde{g}(i, \alpha) = \gamma_{i,j}$. Thus, in order for the verifier to not reject, a cheating prover must send $\tilde{g} \neq g$. By the Schwartz–Zippel lemma, since g and $\tilde{g}$ are low-degree polynomials (provided that the degree d and the number of instances being reduced t are small with respect to $|\mathbb{F}|$), the probability that the uniformly chosen α and r^* are such that $\tilde{g}(\alpha, r^*) = g(\alpha, r^*)$ is small. Whenever $\tilde{g}(\alpha, r^*) \neq g(\alpha, r^*)$ we have that

$$\sum_{\alpha_{j+2},\ldots,\alpha_n \in \{0,1\}} p(r'_1, \ldots, r'_j, r^*, \alpha_{j+2}, \ldots, \alpha_n) = g(\alpha, r^*) \neq \tilde{g}(\alpha, r^*) = \gamma_{j+1}\,,$$

and so the resulting statement $\mathbb{z}' := ((r'_1, \ldots, r'_j, r^*), \gamma_{j+1})$ is false.

3.2.2.2 Defining interactive reducibility

We define interactive reducibility, which captures the capability of merging multiple transcripts/instances into a single transcript/instance while preserving correctness and soundness.

Definition 1. *An ℓ-round public-coin protocol $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ where $\mathbf{V}_{\mathsf{IR}}$ runs in polynomial time is an* **interactive reduction** *for a relation $\mathcal{R}$ with k predicates and soundness error ε if there exists a sequence of predicates $f_0, f_1, \ldots, f_{\mathsf{k}}$ such that the following holds.*

- **Completeness:** *For every $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $j \in [\mathsf{k}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 1$ for every $i \in [\mathsf{t}]$, it holds that:*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] = 1\,.$$

- **Soundness:** *For every $\mathbb{x} \notin L(\mathcal{R})$, $j \in [\mathsf{k}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, if there exists $i \in [\mathsf{t}]$ where $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 0$ then for every (computationally unbounded) $\tilde{\mathbf{P}}_{\mathsf{IR}}$ it holds that:*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] \leq \varepsilon(\mathbb{x}, \mathsf{t})\,.$$

- **Relation identity:** *$f_0(\mathbb{x}, \mathbb{z}) = 1$ if and only if $\mathbb{x} \in L(\mathcal{R})$.*
- **Triviality:** *$f_{\mathsf{k}}(\mathbb{x}, \mathbb{z})$ can be computed in time* $\mathrm{poly}(|\mathbb{x}|, |\mathbb{z}|)$.

We call $\mathbb{x}$ the base instance *and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$* round instances.

An interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ has (polynomially) **bounded output length** if there exists $c \in \mathbb{N}$ such that, for every base instance $\mathbb{x}$, witness $\mathbb{w}$, and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, the new instance $\mathbb{z}'$ output by $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ on inputs $(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}})$ has length at most $|\mathbb{x}|^c$.

3.2.2.3 IOPs from interactive reducibility

We show that any relation with an ℓ-round interactive reduction with k predicates (and bounded output length) has a $(\ell \cdot \mathsf{k})$-round public-coin IOP with query complexity $O(\mathsf{k})$. This is a variation of the protocol described in Section 3.2.1, adapted to work with interactive reducibility. For simplicity, in this overview, we present the protocol only for the case $\ell = 1$ (such as the sumcheck protocol).

To aid with notation, in the description of the protocol, we replace the set S_i (which in Section 3.2.1.1 contained the set of all transcripts generated up until the i-th iteration) with an array A_i where $A_i[j]$ contains all of the round instances generated in the i-th iteration that are associated with the j-th predicate of the interactive reduction. In iteration i, the interactive reduction will be run k times in parallel, where for every $j \in [\mathsf{k}]$ we run given the round instances stored in $A_{i-1}[j]$.

The protocol. Let $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ be a one-round interactive reduction for $\mathcal{R}$ with k predicates. The IOP prover $\mathbf{P}_{\mathsf{IOP}}$ receives as input an instance $\mathbb{x}$ and witness $\mathbb{w}$, and the IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. For every $i \in \{0,\ldots,2\mathsf{k}\}$, $\mathbf{P}_{\mathsf{IOP}}$ defines the $(\mathsf{k}+1)$-entry array A_i as follows

$$A_i[j] := \begin{cases} \{\perp\} & \text{if } j = 0 \\ \varnothing & \text{if } j \in \{1,\ldots,\mathsf{k}\} \end{cases} .$$

   The set $A_i[j]$ will store all instances corresponding to f_j collected by iteration i of the protocol.
2. For $i = 1,\ldots,2\mathsf{k}$:
   (a) $\mathbf{P}_{\mathsf{IOP}}$ sends A_{i-1} to $\mathbf{V}_{\mathsf{IOP}}$.
   (b) $\mathbf{V}_{\mathsf{IOP}}$ sends a random $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}}$ (this corresponds to a message of $\mathbf{V}_{\mathsf{IR}}$).
   (c) $\mathbf{P}_{\mathsf{IOP}}$ sends $a_{i,j} := \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, A_{i-1}[j-1], \alpha_i)$ for all $j \in [\mathsf{k}]$, and sets $A_i[j] := A_{i-1}[j] \cup \{\mathbb{z}_{i,j}\}$ where $\mathbb{z}_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1], \alpha_i, a_{i,j})$ is the output of the interactive reduction verifier given base instance $\mathbb{x}$, round instances $A_{i-1}[j-1]$, verifier randomness α_i, and prover reply $a_{i,j}$.
3. $\mathbf{P}_{\mathsf{IOP}}$ sends $A_{2\mathsf{k}}$ and, for every $i \in \{0,\ldots,2\mathsf{k}\}$, sends $B_i := A_i$. This concludes the interaction.
4. In the decision phase, $\mathbf{V}_{\mathsf{IOP}}$ is given oracle access to a transcript with the following structure:

$$\left(A_0, \alpha_1, (a_{1,1},\ldots,a_{1,\mathsf{k}}), A_1, \ldots, \alpha_{2\mathsf{k}}, (a_{2\mathsf{k},1},\ldots,a_{2\mathsf{k},\mathsf{k}}), A_{2\mathsf{k}}, (B_0,\ldots,B_{2\mathsf{k}})\right) .$$

   $\mathbf{V}_{\mathsf{IOP}}$ performs the checks below.
   (a) *Subset consistency.* Read the arrays $B_0, B_1, \ldots, B_{2\mathsf{k}}$ in their entirety. For every $i \in [2\mathsf{k}]$ and $j \in \{0,\ldots,\mathsf{k}\}$ check that $B_{i-1}[j] \subseteq B_i[j]$.
   (b) *Transcript consistency.* Sample a random $i \in [2\ell]$. Read the arrays A_{i-1} and A_i sent by $\mathbf{P}_{\mathsf{IOP}}$ and the interaction α_i and $(a_{i,1},\ldots,a_{i,\ell})$.
      i. Check that $A_{i-1} = B_{i-1}$ and $A_i = B_i$.
      ii. For every $j \in [\mathsf{k}]$, check that $A_i[j] = A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$ where

$$\mathbb{z}'_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1], \alpha_i, a_{i,j}) ,$$

      is the output of the interactive reduction verifier given base instance $\mathbb{x}$, round instances $A_{i-1}[j-1]$, verifier randomness α_i, and prover reply $a_{i,j}$. (Reject if $\mathbf{V}_{\mathsf{IR}}$ rejects.)
   (c) *Final predicate holds.* Check that $f_{\mathsf{k}}(\mathbb{x}, \mathbb{z}) = 1$ for every $\mathbb{z} \in B_{2\mathsf{k}}[\mathsf{k}]$.

Analysis. The protocol has perfect completeness and soundness error $\max\{\frac{1}{2}, \binom{2\cdot\mathsf{k}}{\mathsf{k}} \cdot \mathsf{k} \cdot \epsilon\}$ where k is the number of predicates and ϵ is the soundness of the interactive reduction respectively. This can be shown in a similar manner to that described in Section 3.2.1.1. The main difference between the two protocols is in the analysis of the proof length. In the protocol of Section 3.2.1.1 the number of sent transcripts doubled in each round. In contrast, in the new protocol, one round instance is added for each predicate. In more detail, for every $i \in [2\mathsf{k}]$ and $j \in [\mathsf{k}]$, we have $|A_i[j]| =$

$|A_{i-1}[j]| + 1$. Since $\mathsf{k} = \text{poly}(|\mathbb{x}|)$, the total number of round instances generated and sent is polynomial in $|\mathbb{x}|$. If the interactive reduction has bounded output length, then each of these round instances has polynomially-bounded length. We can therefore conclude that the overall proof length is $\text{poly}(|\mathbb{x}|)$.

3.2.2.4 Relations with interactive reducibility

Several relations of interest have interactive reductions.

General IPs. We show that any relation with a k-round interactive proof has an m-round interactive reduction with k/m predicates (for any m that divides k). To see this, consider round instances $\mathbb{z}$ that are sets of j-message partial transcripts of the IP. The interactive reduction advances each of the transcripts in the set $\mathbb{z}$ by m rounds, as in the IP. The predicates are defined with respect to the "state function" of the IP, which roughly denotes whether the prover has an accepting strategy with respect to this transcript or whether no strategy will cause the verifier to accept with high probability (over the remaining interaction).

Notice that if this interactive reduction is used in the protocol of Section 3.2.2.3, this yields the protocol described in Section 3.2.1.1. This interactive reduction does not have bounded output length since the new round instance stores all of the previous transcripts and their continuations. Therefore the resulting IOP does not achieve $O(1)$ total query complexity.

Sumcheck protocol. Using the ideas described in Section 3.2.2.2, we show that any relation that can be reduced into k-variate sumcheck has a one-round interactive reduction with k predicates and *bounded output length*. As a result, any relation that can be reduced into k-variate sumcheck has a $O(\mathsf{k})$-round public-coin IOP with query complexity $O(1)$.

Shamir's protocol. Shamir's protocol [Sha92] gives an IP for all of PSPACE, thereby showing the IP = PSPACE theorem. Extending the ideas developed in Section 3.2.2.2, we show a one-round interactive reduction with bounded output length and a polynomial number of predicates for Shamir's protocol. This establishes that every language in PSPACE has a $\text{poly}(|\mathbb{x}|)$-round public-coin IOP with query complexity $O(1)$.

Future directions. We leave the exploration of what other relations have interactive reductions to future work. Following the extensive use of polynomials in both the sumcheck protocol and Shamir's protocol, it seems likely that these techniques can be adapted to also work for low-depth circuits through the delegation protocol in [GKR15].

3.2.3 Towards transforming IPs to IOPs

We discuss the problem of transforming IPs into IOPs. We begin by describing a solution in [Dru11a] that transforms a single-round IP into a single-round IOP. Following that, we describe the challenges of extending this approach to work for multi-round IPs.

3.2.3.1 The case of a single-round IP

The case of a single-round was settled by Drucker [Dru11a], whose work implies a transformation from a public-coin single-round IP to a single-round IOP where the verifier reads $O(1)$ bits from the communication transcript (here consisting of the prover message and the verifier message). His construction uses as building blocks the randomness-efficient amplification technique of [BGG90] and *PCPs of proximity* (PCPPs) [DR04; BGHSV06b].[5] We give a high-level overview of his construction.

In a public-coin single-round IP, given a common input instance $\mathbb{x}$, the verifier $\mathbf{V}_{\mathsf{IP}}$ sends randomness α, the prover $\mathbf{P}_{\mathsf{IP}}$ sends a message a, and the verifier $\mathbf{V}_{\mathsf{IP}}$ decides whether to accept by applying a predicate to $(\mathbb{x}, \alpha, a)$. Consider the non-deterministic machine M such that $M(\mathbb{x}, \alpha) = 1$ if and only if there exists a such that $\mathbf{V}_{\mathsf{IP}}$ accepts $(\mathbb{x}, \alpha, a)$. The constructed IOP works as follows:

1. the IOP verifier sends $\mathbf{V}_{\mathsf{IP}}$'s randomness α;
2. the IOP prover computes $\mathbf{P}_{\mathsf{IP}}$'s message a and produces a PCPP string Π for the claim "$M(\mathbb{x}, \alpha) = 1$";
3. the IOP verifier checks Π using the PCPP verifier with explicit inputs M and $\mathbb{x}$ and implicit input α.

This IOP is sound if the underlying IP is "randomness-robust", which means that if $\mathbb{x}$ is not in the language then with high probability over α it holds that α is *far* from any accepting input for $M(\mathbb{x}, \cdot)$. Drucker achieves this property by using an amplification technique in [BGG90] that achieves soundness error $2^{-|\alpha|}$ while using $O(|\alpha|)$ random bits (standard amplification would, when starting with a constant-soundness protocol, result in $\omega(|\alpha|)$ random bits). Thus, with high probability, α is not only a "good" random string (which holds for any single-round IP) but also is δ-far from any "bad" random string, for some small constant $\delta > 0$. This follows since the ball of radius δ around any bad random string has size $2^{\delta'|\alpha|}$, for some small constant δ' that depends only on δ.

3.2.3.2 Challenges of extending the single-round approach to multi-round IPs

We wish to obtain a similarly efficient transformation for a public-coin k-round IP where $\mathsf{k} = \mathrm{poly}(n)$.

One possible approach would be to reduce the number of rounds of the given IP from k to 1 and then apply the transformation for single-round IPs. The round reduction of Babai and Moran [BM88] shows that any public-coin k-round IP can be transformed into a one-round IP where efficiency parameters grow by $n^{O(\mathsf{k})}$. This transformation, however, is not efficient for super-constant values of k. Moreover, it

[5] A PCPP is a PCP system where the verifier has oracle access to its input in addition to the prover's proof; the soundness guarantee is that if the input is *far* (in Hamming distance) from any input in the language, then the verifier accepts with small probability.

is undesirable even when k is constant because the transformation overhead is not a fixed polynomial (the exponent depends on k rather than being a fixed constant).

Therefore, we seek an approach that directly applies to a multi-round IP. Unfortunately, Drucker's approach for one-round IPs does not generalize to multiple-round IPs for several reasons. First, the corresponding machine $M(\mathbb{x}, \alpha_1, \ldots, \alpha_\mathsf{k})$ (which accepts if and only if there exist prover messages $a_1, \ldots, a_\mathsf{k}$ such that $\mathbf{V}_{\mathsf{IP}}$ accepts $(\mathbb{x}, \alpha_1, a_1, \ldots, \alpha_\mathsf{k}, a_\mathsf{k})$) does not capture the soundness of the interactive proof because it fails to capture interaction (a protocol may be sound according to the IP definition and, yet, for every $\mathbb{x}$ and $\alpha_1, \ldots, \alpha_\mathsf{k}$ it could be that $M(\mathbb{x}, \alpha_1, \ldots, \alpha_\mathsf{k}) = 1$). Moreover, it is not clear how to perform a randomness-efficient amplification for multiple rounds that makes the protocol sufficiently "randomness robust" for the use of a PCPP. The main reason is that to get soundness error 2^{-m} (as in [Dru11a]), the techniques of [BGG90] add $O(m)$ bits *per round*, which is too much when the protocol has many rounds (see Section 3.2.4.3 for a more detailed discussion on why this approach fails for many rounds).

We give a different solution that circumvents this step and works for any number of rounds. Our transformation from k-round IP to an IOP in two stages. In the first stage, we transform the IP into one in which the verifier reads only $O(1)$ bits from each random message it sends. In the second stage, we transform the IP into an IOP with $O(1)$ per-round query complexity, simultaneously for each prover message and each verifier message. We achieve this via a new notion of PCPs that we call *index-decodable PCPs*, and we describe in Section 3.2.5. First, we explain how to achieve the property that the verifier reads $O(1)$ bits from each of its random messages to the prover.

3.2.4 Local access to randomness

We transform a public-coin IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ into an IP $(\mathbf{P}'_{\mathsf{IP}}, \mathbf{V}'_{\mathsf{IP}})$ whose verifier (i) reads $O(1)$ bits from each of its random messages to the prover, and (ii) has logarithmic decision randomness (the randomness used by the verifier in the post-interaction decision stage). For now, the verifier reads in full every message received from the prover, and only later we discuss how to reduce the query complexity to prover messages while preserving the query complexity to the verifier random messages. Note that in the full transformation, we need to support round-query IOPs, but we omit the changes required for this for simplicity of this overview.

3.2.4.1 One-round public-coin proofs

In order to describe our ideas we begin with the simple case of one-round public-coin interactive proofs. Recall from Section 3.2.3 that this case is solved in [Dru11a], but we nevertheless first describe our alternative approach for this case and after that we will discuss the multiple-round case.

A strawman protocol. Recall that in a one-round public-coin IP the verifier sends

a uniformly random message, the prover replies with some answer, and the verifier uses both of these messages to decide whether to accept. An idea to allow the verifier to not read in full its own random message would be for the prover to send the received random message back to the verifier, and the verifier to use this latter and test consistency with its own randomness. Given an instance $\mathbb{x}$: $\mathbf{V}'_{\mathsf{IP}}$ sends $\mathbf{V}_{\mathsf{IP}}$'s random message $\alpha \in \{0,1\}^{\mathsf{r}}$; $\mathbf{P}'_{\mathsf{IP}}$ replies with $\alpha' := \alpha$ and the message $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \alpha)$; and $\mathbf{V}'_{\mathsf{IP}}$ checks that α and α' agree on a random location and that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha', a) = 1$.

This new IP is complete, and its verifier queries its random message at one location to conduct the consistency test. However, the protocol might not be sound, as we explain. Suppose that $\mathbb{x} \notin L$. Let r be the length of α, β be the soundness error of the original IP, $\delta \in (0,1)$ be a small constant to be specified later, and let $\nu_{\mathsf{r},\delta}$ be the volume of the Hamming sphere of radius $\mathsf{r} \cdot \delta$ in $\{0,1\}^{\mathsf{r}}$. A choice of verifier message α is *bad* if there exists a such that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha, a) = 1$. By the soundness guarantee of $\mathbf{V}_{\mathsf{IP}}$, the fraction of bad choices of random verifier messages is at most β. A choice of verifier message α is *ball-bad* if there exist a bad α' that is δ-close to α. By the union bound, the fraction of ball-bad coins is at most $\gamma = \beta \cdot \nu_{\mathsf{r},\delta}$.

Let E be the event over the choice of α that the prover sends α' that is δ-far from α.

- Conditioned on E occurring, $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least δ (whenever $\mathbf{V}'_{\mathsf{IP}}$ chooses a location on which α and α' disagree).
- Conditioned on E not occurring, $\mathbf{P}'_{\mathsf{IP}}$ cannot send any α' and a such that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha', a) = 1$ unless α is ball-bad, and so $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least $1 - \gamma$.

Therefore, for the new IP to be sound, we need $\gamma = \beta \cdot \nu_{\mathsf{r},\delta}$ to be small. Notice that $\nu_{\mathsf{r},\delta} = 2^{H(\delta)\cdot \mathsf{r}}$ depends on r but not on β (here H is the entropy function $H(\delta) := -\delta \log \delta - (1-\delta)\log(1-\delta)$). Thus we need to achieve $\log 1/\beta > H(\delta) \cdot \mathsf{r}$. As in Drucker's transformation, this can be done using the randomness-efficient soundness amplification of [BGG90], *but we deliberately take a different approach that will generalize for multiple rounds*.

Shrinking γ using extractors. Let Ext be an extractor with output length r, seed length $O(\log \mathsf{r} + \log 1/\beta)$, and error β;[6] such extractors are constructed in [GUV09]. Assume that $\beta = 1/O(\mathsf{r})$, which can be achieved using $O(\log \mathsf{r})$ parallel repetitions, and so the seed length is $O(\log 1/\beta)$. Suppose that the prover and verifier have access to a sample z from a source D with high min-entropy. Consider the following IP: $\mathbf{V}'_{\mathsf{IP}}$ sends s; $\mathbf{P}'_{\mathsf{IP}}$ replies with $s' := s$ and $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s'))$; $\mathbf{V}'_{\mathsf{IP}}$ checks that s and s' agree on a random location and that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s'), a) = 1$.

At most a 2β-fraction of the seeds s are such that there exists a such that

$$\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s), a) = 1\,,$$

because Ext is an extractor with error β and D is a distribution with high min-entropy. By an identical argument to the one done previously, either $\mathbf{P}'_{\mathsf{IP}}$ sends s' that is far from s and so $\mathbf{V}'_{\mathsf{IP}}$ rejects with constant probability, or $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least

[6] A function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-extractor if, for every random variable X over $\{0,1\}^n$ with min-entropy at least k, the statistical distance between $\mathsf{Ext}(X, U_d)$ and U_m is at most ε.

$\gamma = 2\beta \cdot \nu_{\mathsf{r}',\delta}$ where $\mathsf{r}' = |s| = O(\log 1/\beta)$. Thus we have that $\gamma = 2 \cdot \beta \cdot 2^{H(\delta)\cdot O(\log 1/\beta)}$. We can now set δ to be a small enough constant such that $\gamma = O(\sqrt{\beta})$.

Generating a source of high min-entropy. We describe how the prover and verifier can agree on a sample from a high-entropy source by leveraging the following observation: if z is a uniformly random string and z' is an arbitrary string that is close in Hamming distance to z, then z' has high min-entropy. Thus we can sample via similar ideas as above: $\mathbf{V}'_{\mathsf{IP}}$ samples and sends z; $\mathbf{P}'_{\mathsf{IP}}$ replies with $z' := z$; and $\mathbf{V}'_{\mathsf{IP}}$ checks that z and z' agree on a random location. (So $\mathbf{V}'_{\mathsf{IP}}$ reads one bit of its random message z.) If, with constant probability over z, $\mathbf{P}'_{\mathsf{IP}}$ sends z' that is far from z, then $\mathbf{V}'_{\mathsf{IP}}$ rejects with constant probability. Otherwise, we show that z' has high min-entropy because with high probability it agrees with z on most of its locations.

Putting it all together. Let $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ be a public-coin single-round IP with soundness error β and randomness complexity r, and let Ext be an extractor with output length r, seed length $O(\log 1/\beta)$, and error β. The new IP $(\mathbf{P}'_{\mathsf{IP}}, \mathbf{V}'_{\mathsf{IP}})$ is as follows.

- *Sample high min-entropy source:* $\mathbf{V}'_{\mathsf{IP}}$ sends z and $\mathbf{P}'_{\mathsf{IP}}$ replies with $z' := z$.
- *Sample extractor seed:* $\mathbf{V}'_{\mathsf{IP}}$ sends s and $\mathbf{P}'_{\mathsf{IP}}$ replies with $s' := s$.
- *Prover message:* $\mathbf{P}'_{\mathsf{IP}}$ sends $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z', s'))$.
- *Verification:* $\mathbf{V}'_{\mathsf{IP}}$ checks that z and z' agree on a random location, s and s' agree on a random location, and $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z', s'), a) = 1$.

3.2.4.2 Extending to multiple rounds

In order to extend the previously described protocol to multiple rounds, we leverage the notion of *round-by-round soundness*. An IP for a language L has round-by-round soundness error β_{rbr} if there exists a "state" function such that: (i) for $\mathbb{x} \notin L$, the starting state is "doomed"; (ii) for every doomed state and next message that a malicious prover might send, with probability β_{rbr} over the verifier's next message, the protocol state will remain doomed; (iii) if at the end of interaction the state is doomed then the verifier rejects.

In the analysis of the one-round case there was an event (called *bad*) over the IP verifier's random message α such that if this event does not occur then the prover has no accepting strategy. This event can be replaced, in the round-by-round case, by the event that, in a given round, the verifier chooses randomness where the transcript remains doomed. This idea leads to a natural extension of the one-round protocol described in Section 3.2.4.1 to the multi-round case, which is our final protocol.

Let $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ be a public-coin k-round IP with round-by-round soundness error β_{rbr} and randomness complexity r, and Ext an extractor with output length r, seed length $O(\log 1/\beta_{\mathrm{rbr}})$, and error β_{rbr}.

- For each round $j \in [\mathsf{k}]$ of the original IP:
  1. *Sample high min-entropy source:* $\mathbf{V}'_{\mathsf{IP}}$ sends z_j and $\mathbf{P}'_{\mathsf{IP}}$ replies with $z'_j := z_j$.
  2. *Sample extractor seed:* $\mathbf{V}'_{\mathsf{IP}}$ sends s_j and $\mathbf{P}'_{\mathsf{IP}}$ replies with $s'_j := s_j$.
  3. *Prover message:* $\mathbf{P}'_{\mathsf{IP}}$ sends $a_j := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, \ldots, \alpha_j)$ where $\alpha_i := \mathsf{Ext}(z_i, s_i)$.

- $\mathbf{V}'_{\mathsf{IP}}$ accepts if and only if the following tests pass:
  1. Choose a random location and, for every $j \in [\mathsf{k}]$, test that z_j and z'_j agree on this location.
  2. Choose a random location and, for every $j \in [\mathsf{k}]$, test that s_j and s'_j agree on this location.
  3. For every $j \in [\mathsf{k}]$, compute $\alpha_j := \mathsf{Ext}(z'_j, s'_j)$. Check that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, a_1, \ldots, \alpha_{\mathsf{k}}, a_{\mathsf{k}}) = 1$.

The soundness analysis of this protocol is similar to the one-round case. Suppose that $\mathbb{x} \notin L$. Then the empty transcript is "doomed". By an analysis similar to the one-round case, except where we set "bad" verifier messages to be ones where the transcript state switches from doomed to not doomed, if a round begins with a doomed transcript then except with probability $\gamma = O(\sqrt{\beta_{\text{rbr}}})$ the transcript in the next round is also doomed. Thus, by a union bound, the probability that the transcript ends up doomed, and as a result the verifier rejects, is at least $1 - O(\mathsf{k} \cdot \sqrt{\beta_{\text{rbr}}})$. As shown in [CCHLRR18] round-by-round soundness error can be reduced via parallel repetition, albeit at a lower rate than regular soundness error. Thus, by doing enough parallel repetition before applying our transformation, the round-by-round soundness error β_{rbr} can be reduced enough so that the verifier rejects with constant probability.

The above protocol has $2\mathsf{k}_{\mathsf{IP}}$ rounds. The verifier reads 1 bit from each of its random messages, and has $O(\log |\mathbb{x}|)$ bits of decision randomness (to sample random locations for testing consistency between each z'_j and z_j and between each s'_j and s_j). To achieve k_{IP} rounds, we first apply the round reduction of [BM88] on the original IP to reduce to $\mathsf{k}_{\mathsf{IP}}/2$ rounds, and then apply our transformation.

3.2.4.3 Why randomness-efficient soundness amplification is insufficient

We briefly sketch why applying randomness-efficient soundness amplification in the style of [BGG90] is insufficient in the multi-round case, even if we were to consider round-by-round soundness. Recall that we wish for $\beta_{\text{rbr}} \cdot 2^{\Theta(\mathsf{r})}$ to be small, where β_{rbr} is the round-by-round soundness of the protocol and r is the number of random bits sent by the verifier in a single round. Bellare, Goldreich and Goldwasser [BGG90] show that, starting with constant soundness and randomness r, one can achieve soundness error 2^{-m} using $\mathsf{r}' = O(\mathsf{r} + m)$ random bits; they do this via m parallel repetitions where the randomness between repetitions is shared in a clever way. Using parallel repetition, achieving round-by-round soundness error 2^{-m} requires m/k repetitions (see [CCHLRR18]). Thus, even if we were to show that the transformation of [BGG90] reduces round-by-round soundness error at the same rate as standard parallel repetition (as it does for standard soundness), in order to get round-by-round soundness error 2^{-m}, we would need $\mathsf{r}' = O(\mathsf{r} + m \cdot \mathsf{k})$ bits of randomness. This would achieve $\beta_{\text{rbr}} \cdot 2^{\Theta(\mathsf{r}')} = 2^{-m} \cdot 2^{\Theta(\mathsf{r}+m\mathsf{k})}$, which, for super-constant values of k, is greater than 1 regardless of r.

3.2.5 Index-decodable PCPs

We introduce *index-decodable PCPs*, a notion of PCP that works on *multi-indexed relations*. A multi-indexed relation $\mathcal{R}$ is a set of tuples $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$ where $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}])$ is the index vector, $\mathbb{x}$ the instance, and $\mathbb{w}$ the witness. As seen in the following definition, an index-decodable PCP treats the index vector $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}])$ and the instance $\mathbb{x}$ differently, which is why they are not "merged" into an instance $\mathbb{x}' = (\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x})$ (and why we do not consider standard relations).

Definition 2. *An* **index-decodable PCP** *for a multi-indexed relation* $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ *is a tuple of algorithms* $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$*, where* $\mathbf{I}_{\mathsf{PCP}}$ *is the (honest) indexer,* $\mathbf{P}_{\mathsf{PCP}}$ *the (honest) prover,* $\mathbf{V}_{\mathsf{PCP}}$ *the verifier,* $\mathbf{iD}_{\mathsf{PCP}}$ *the index decoder, and* $\mathbf{wD}_{\mathsf{PCP}}$ *the witness decoder. The system has (perfect completeness and)* decodability bound κ_{PCP} *if the following conditions hold.*

- ***Completeness.*** *For every* $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$*,*

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

- ***Decodability.*** *For every* $\mathbb{x}$*, indexer proofs* $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$*, and malicious prover proof* $\tilde{\Pi}$*, if*

$$\Pr_{\alpha}\left[\ \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}}(\mathbb{x};\alpha) = 1\ \right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)$$

  then $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in \mathcal{R}$*.*

The indexer $\mathbf{I}_{\mathsf{PCP}}$ separately encodes each index, independent of indices and the instance, to obtain a corresponding *indexer proof*. The prover $\mathbf{P}_{\mathsf{PCP}}$ gets all the data as input (index vector, instance, and witness) and outputs a *prover proof*. The verifier $\mathbf{V}_{\mathsf{PCP}}$ gets the instance as input and has query access to $\mathsf{k}+1$ oracles (k indexer proofs and 1 prover proof), and outputs a bit.

The decodability condition warrants some discussion. The usual soundness condition of a PCP for a standard relation $\mathcal{R}$ has the following form: "if $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\Pi}}(\mathbb{x})$ accepts with high-enough probability then there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$". For a multi-indexed relation it could be that for any given instance $\mathbb{x}$ there exist indexes $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}]$ and a witness $\mathbb{w}$ such that $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Since we do not trust the indexer's outputs, a soundness condition is not meaningful.

Instead, the decodability condition that we consider has the following form: "if $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}}(\mathbb{x})$ accepts with high-enough probability then $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ where $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}]$ and $\mathbb{w}$ are the decoded indices and witness respectively found in $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$ and $\tilde{\Pi}$". It is crucial that the index decoder receives as input the relevant indexer proof but not also the instance, or else the decodability condition would

be trivially satisfied (the index decoder could output the relevant index of the lexicographically first index vector putting the instance in the relation). This ensures that the proofs collectively convince the verifier not only that there exists an index vector and witness that place the instance in the relation, but that the prover encoded a witness that, along with index vector obtainable from the index oracles via the index decoder, places the instance in the relation.

We do not require the indexer or the decoders to be efficient. However, in some applications, it is useful to have an efficient indexer and decoders, and indeed we construct an index-decodable PCP with an efficient indexer and decoders.

Remark 3.2.3 (comparison with holography)**.** We compare index-decodable PCPs and holographic PCPs, which also work for indexed relations (see [CHMMVW20] and references therein). In both cases, an indexer produces an encoding of the index (independent of the instance). However, there are key differences between the two: (i) in an index-decodable PCP the indexer works separately on each entry of the index vector, while in a holographic PCP there is a single index; moreover, (ii) in a holographic PCP the indexer is trusted in the sense that security is required to hold only when the verifier has oracle access to the honest indexer's output, but in an index-decodable PCP, the indexer is **not trusted** in the sense that the malicious prover can choose encodings for all of the indices. *Both differences are essential properties for our transformation of IPs into IOPs.*

We construct a binary index-decodable PCPs with $O(1)$ query complexity per oracle.

Theorem 3. *Any multi-indexed relation* $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ *to which membership can be verified in nondeterministic time* T *has a non-adaptive index-decodable PCP with the following parameters:*

Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length (per proof)	$O(\lvert\mathbb{i}[i]\rvert)$
Prover proof length	$\mathrm{poly}(T)$
Alphabet size	2
Queries per oracle	$O(1)$
Randomness	$O(\log \lvert\mathbb{x}\rvert)$
Decodability bound	$O(1)$
Indexer running time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Prover running time	$\mathrm{poly}(T)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \log T)$
Index decoding running	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Witness decoding time	$\mathrm{poly}(T)$

Our construction achieves optimal parameters similar to the PCP theorem: it has $O(1)$ query complexity (per oracle) over a binary alphabet, and the randomness complexity is logarithmic, *independent* of the number of indexes k. Achieving small randomness complexity is challenging and useful. First, it facilitates proof composition

(where a prover writes a proof for every possible random string), which is common when constructing zero-knowledge PCPs (e.g., [IW14]). Second, small randomness complexity is necessary for our hardness of approximation results.

A similar notion is (implicitly) considered in [ALMSS98] but their construction does not achieve the parameters we obtain in Theorem 3 (most crucially, they do not achieve small randomness).

3.2.6 Local access to prover messages

We show how to transform an IP into an IOP by eliminating the need of the verifier to read more than a few bits of each prover message. Note that in the full transformation, we need to support round-query IOPs, but we omit the changes required for this for simplicity of this overview. This transformation preserves the number of bits read by the verifier to its own interaction randomness. Thus, combining it with the transformation described in Section 3.2.4, this completes the proof (overview) of Theorem 1.

We transform any public-coin IP into an IOP by using an index-decodable PCP. In a public-coin k-round IP, the prover $\mathbf{P}_{\mathsf{IP}}$ and verifier $\mathbf{V}_{\mathsf{IP}}$ receive as input an instance $\mathbb{x}$ and then, in each round i, the verifier $\mathbf{V}_{\mathsf{IP}}$ sends randomness α_i and the prover replies with a message $a_i \leftarrow \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, \ldots, \alpha_i)$; after the interaction, the verifier $\mathbf{V}_{\mathsf{IP}}$ runs an efficient probabilistic algorithm with decision randomness α_{dc} on the transcript $(\mathbb{x}, \alpha_1, a_1, \ldots, \alpha_{\mathsf{k}}, a_{\mathsf{k}})$ to decide whether to accept or reject.

The IP verifier $\mathbf{V}_{\mathsf{IP}}$ defines a multi-indexed relation $\mathcal{R}(\mathbf{V}_{\mathsf{IP}})$ consisting of tuples

$$\left(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}', \mathbb{w}\right) = \left(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}), \bot\right)$$

such that the IP verifier $\mathbf{V}_{\mathsf{IP}}$ accepts the instance $\mathbb{x}$, transcript $(\alpha_1, a_1, \ldots, \alpha_{\mathsf{k}}, a_{\mathsf{k}})$, and decision randomness α_{dc}. (Here we do not rely on witnesses although the definition of index-decodable PCPs supports this.)

From IP to IOP. Let $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be an index-decodable PCP for the relation $\mathcal{R}(\mathbf{V}_{\mathsf{IP}})$. We construct the IOP as follows. The IOP prover and IOP verifier receive an instance $\mathbb{x}$. In round $i \in [\mathsf{k}]$, the IOP verifier sends randomness α_i (just like the IP verifier $\mathbf{V}_{\mathsf{IP}}$) and the (honest) IOP prover sends the indexer proof $\pi_i := \mathbf{I}_{\mathsf{PCP}}(a_i)$ where $a_i \leftarrow \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, \ldots, \alpha_i)$. In a final additional message (which can be sent at the same time as the last indexer proof π_{k}), the IOP prover sends $\Pi := \{\Pi_{\alpha_{\mathsf{dc}}}\}_{\alpha_{\mathsf{dc}}}$ where, for every possible choice of decision randomness α_{dc}, $\Pi_{\alpha_{\mathsf{dc}}}$ is an index-decodable PCP prover proof to the fact that $(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}), \bot) \in \mathcal{R}(\mathbf{V}_{\mathsf{IP}})$. After the interaction, the IOP verifier samples IP decision randomness α_{dc} and checks that $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}_{\alpha_{\mathsf{dc}}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}, \alpha_{\mathsf{dc}}) = 1$.

Proof sketch. Completeness follows straightforwardly from the construction. We now sketch a proof of soundness. Letting L be the language decided by $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$, fix an instance $\mathbb{x} \notin L$ and a malicious IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. Given interaction randomness

$\alpha_1, \ldots, \alpha_k$, consider the messages $\tilde{\pi}_1, \ldots, \tilde{\pi}_k$ output by $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ in the relevant rounds ($\tilde{\pi}_i$ depends on $\alpha_1, \ldots, \alpha_i$) and the message $\tilde{\Pi} = \{\tilde{\Pi}_{\alpha_{dc}}\}_{\alpha_{dc}}$ output by $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ in the last round (this message depends on $\alpha_1, \ldots, \alpha_k$). We consider two complementary options of events over the IOP verifier's randomness $(\alpha_1, \ldots, \alpha_k, \alpha_{dc})$.

1. With high probability the proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_k$ and $\tilde{\Pi}_{\alpha_{dc}}$ generated while interacting with $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ using randomness $\alpha_1, \ldots, \alpha_k$ and α_{dc} are such that

$$\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_k), (\mathbb{x}, \alpha_1, \ldots, \alpha_k, \alpha_{dc}), \perp\right) \notin \mathcal{R}(\mathbf{V}_{\mathsf{IP}}) .$$

   If this is true, then, by the decodability property of the index-decodable PCP, the IOP verifier must reject with high probability over the choice of randomness for $\mathbf{V}_{\mathsf{PCP}}$.

2. With high probability the proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_k$ and $\tilde{\Pi}_{\alpha_{dc}}$ generated while interacting with $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ using randomness $\alpha_1, \ldots, \alpha_k$ and α_{dc} are such that

$$\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_k), (\mathbb{x}, \alpha_1, \ldots, \alpha_k, \alpha_{dc}), \perp\right) \in \mathcal{R}(\mathbf{V}_{\mathsf{IP}}) .$$

   We prove that this case cannot occur by showing that it contradicts the soundness of the original IP. Suppose towards contradiction that the above is true. We use $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ and the index decoder of the index-decodable PCP, $\mathbf{iD}_{\mathsf{PCP}}$, to construct a malicious IP prover for the original IP as follows.

   In round i, the transcript $(\alpha_1, a_1, \ldots, \alpha_{i-1}, a_{i-1})$ has already been set during previous interaction. The IP verifier sends randomness α_i. The IP prover sends $a_i := \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_i)$ to the IP verifier, where $\tilde{\pi}_i := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1, \ldots, \alpha_i)$. Recall that

$$\left(\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_k), (\mathbb{x}, \alpha_1, \ldots, \alpha_k, \alpha_{dc}), \perp\right) \in \mathcal{R}(\mathbf{V}_{\mathsf{IP}}) ,$$

   if and only if the IP verifier accepts given instance $\mathbb{x}$, randomness $(\alpha_1, \ldots, \alpha_k, \alpha_{dc})$, and prover messages $\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_k)$, which is precisely what the IP prover supplies it with. Since the event that

$$\left(\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_k), (\mathbb{x}, \alpha_1, \ldots, \alpha_k, \alpha_{dc}), \perp\right) \in \mathcal{R}(\mathbf{V}_{\mathsf{IP}}) ,$$

   happens with high probability, this implies that with high probability the IP verifier will accept, contradicting soundness of the original IP. Here we crucially used the fact that the decoder $\mathbf{D}_{\mathsf{PCP}}$ does not depend on the instance of the index-decodable PCP (which consists of $\mathbb{x}$ and all of the IP verifier's random coins $\alpha_1, \ldots, \alpha_k, \alpha_{dc}$) or on the other indexer messages.

The resulting IOP has k rounds, exactly as in the original IP. The IOP verifier uses as much randomness as the original IP verifier with the addition of the randomness used by the index-decodable PCP. The query complexity is that of the underlying

verifier of the index-decodable PCP. The proof length and alphabet are the same as those of the index-decodable PCP.

Preserving local access to randomness. The transformation described above can be modified to preserve the query complexity of the verifier to its own interaction randomness if the verifier is non-adaptive with respect to its queries to its random messages (i.e., the choice of bits that it reads depends only on $\mathbb{x}$ and α_{dc}). We can redefine the multi-indexed relation $\mathcal{R}(\mathbf{V}_{\mathsf{IP}})$ to have as explicit inputs the instance $\mathbb{x}$, decision randomness α_{dc}, and the bits of $\alpha_1, \ldots, \alpha_{\mathsf{k}}$ that the verifier needs to read to decide whether to accept or reject (rather than the entire interaction randomness strings). In more detail, suppose that the verifier reads q bits from its own interaction randomness. Then the new multi-indexed relation consists of tuples:

$$\left(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}', \mathbb{w}\right) = \left(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, b_1, \ldots, b_{\mathsf{q}}, \alpha_{\mathsf{dc}}), \bot\right)$$

such that given decision randomness α_{dc} the IP verifier $\mathbf{V}_{\mathsf{IP}}$ accepts given instance $\mathbb{x}$, decision randomness α_{dc}, prover messages $(a_1, \ldots, a_{\mathsf{k}})$, and $(b_1, \ldots, b_{\mathsf{q}})$ as answers to its q queries to $\alpha_1, \ldots, \alpha_{\mathsf{k}}$.

Given a multi-indexed PCP for this relation, the IP to IOP transformation is identical to the one described above, except that after the interaction, the IOP verifier samples IP decision randomness, queries its own interaction randomness to get answers $b_1, \ldots, b_{\mathsf{q}}$, and these replace $\alpha_1, \ldots, \alpha_{\mathsf{k}}$ as explicit inputs to the index-decodable PCP verifier $\mathbf{V}_{\mathsf{PCP}}$.

3.2.7 Constructing index-decodable PCPs

We describe how to construct index-decodable PCPs: in Section 3.2.7.1 we outline a randomness-efficient index-decodable PCP that makes $O(1)$ queries to each of its oracles, where the indexer proofs are over the binary alphabet and the prover proof is over a large alphabet; then in Section 3.2.7.2 we use proof composition to reduce the alphabet size of the latter.

3.2.7.1 Basic construction from PCPPs

We outline a construction of an index-decodable PCP with $O(1)$ query complexity to each indexer proof and to the prover proof, and where the prover proof is over a large alphabet (of size 2^{k}). For a later proof composition while preserving polynomial proof length, here we additionally require that the verifier has logarithmic randomness complexity.

Building blocks. In our construction we rely on variants of PCPPs. Recall that a PCPP is a PCP system where the verifier has oracle access to its input in addition to the prover's proof; the soundness guarantee is that if the input is *far* (in Hamming distance) from any input in the language, then the verifier accepts with small probability.

We use PCPPs that are *multi-input* and *oblivious*. We explain each of these properties.

- A PCPP is multi-input if the verifier has oracle access to multiple (oracle) inputs. The soundness guarantee is that, for every vector of inputs that satisfy the machine in question, if at least one input oracle is far from the respective satisfying input, then the verifier accepts with small probability.
- A (non-adaptive) PCPP is oblivious for a family of nondeterministic machines $\mathcal{M} = \{M_i\}_{i \in [\mathsf{k}]}$ if the queries made by the verifier to its oracles depend only on $\mathcal{M}$ and its randomness. In particular they do not depend on i. This property will be used later to facilitate bundling queries. We will have k PCPs, each with a different M_i, but the verifier will use the same randomness in each test. Since the PCPPs are oblivious, this means that the verifier makes the same queries for every test. Thus we can group together the k proofs into a single proof with larger alphabet and maintain good query complexity on this proof. This property is important in order to achieve our final parameters.

See Section 3.7.1 for definitions for the above notions, and how to obtain them from standard PCPPs. Henceforth, all PCPPs that we use will be over the binary alphabet and have constant proximity, constant soundness error, constant query complexity, and logarithmic randomness complexity.

The construction. We construct an index-decodable PCP for a multi-indexed relation $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ whose membership can be verified efficiently.

The indexer encodes each index via an error-correcting code with (constant) relative distance greater than the (constant) proximity parameter of the PCPP used later. The prover uses PCPPs to prove that there exist indexes and a witness that put the given instance in the relation and adds consistency checks to prove that the indices are consistent with those encoded by the indexer. The verifier checks each of these claims. The index decoder decodes the indexer proofs using the same code.

In slightly more detail, the index-decodable PCP is as follows.

- $\mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$: Encode the index $\mathbb{i}[i]$ as π_i using an error-correcting code.
- $\mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$:
  1. *Encoding the indexes.* Compute Π_*, an encoding of the string $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{w})$.
  2. *Membership of encoding.* Compute a PCPP string Π_{mem} for the claim that $M_*(\mathbb{x}, \Pi_*) = 1$ where M_* checks that Π_* is a valid encoding of indexes and a witness that put $\mathbb{x}$ in $\mathcal{R}$.
  3. *Consistency of encoding.* For every $j \in [\mathsf{k}]$, compute a PCPP string Π_j for the claim that $M_j(\pi_j, \Pi_*) = 1$ where M_j checks that π_j and Π_* are valid encodings and that the string $\mathbb{i}[j]$ encoded within π_j is equal to the matching string encoded within Π_*.
  4. Output $(\Pi_*, \Pi_{\mathsf{mem}}, \Pi_{\mathbb{i}})$ where $\Pi_{\mathbb{i}}$ are the proofs $\Pi_1, \ldots, \Pi_{\mathsf{k}}$ "bundled" together into symbols of k bits such that $\Pi_{\mathbb{i}}[q] = (\Pi_1[q], \ldots, \Pi_{\mathsf{k}}[q])$.

- $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})}(\mathbb{x})$: Check that all the tests below pass.
  1. *Membership.* Run the PCPP verifier on the claim that $M_*(\mathbb{x},\tilde{\Pi}_*)=1$ using proof oracle $\tilde{\Pi}_{\mathsf{mem}}$.
  2. *Consistency.* For every $j\in[\mathsf{k}]$, run the PCPP verifier on the claim that $M_j(\tilde{\pi}_j,\tilde{\Pi}_*)=1$ using proof oracle $\tilde{\Pi}_j$. These k tests are run *with the same randomness*. Since the PCPP is oblivious and randomness is shared, the queries made by the PCPP verifier in each test are identical, and so each query can be made by reading the appropriate k-bit symbols from $\tilde{\Pi}_{\mathbb{i}}$.

- $\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_j)$: output the codeword closest to $\tilde{\pi}_j$ in the error-correcting code.

- $\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})$: Let $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}})$ be the codeword closest to $\tilde{\Pi}_*$ in the error-correcting code and output $\tilde{\mathbb{w}}$.

Completeness follows straightforwardly from the construction. We now sketch decodability.

Decodability. Fix an instance $\mathbb{x}$, indexer proofs $\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}}$, and prover proof $(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})$. Suppose that the verifier accepts with high-enough probability. We argue that this implies that there exists $\mathbb{w}$ such that $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1),\ldots,\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}),\mathbb{x},\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right)\in\mathcal{R}$. Specifically, we argue that $\tilde{\Pi}_*$ encodes indices $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ and witness $\tilde{\mathbb{w}}$ that place $\mathbb{x}$ in $\mathcal{R}$ and, additionally, each $\tilde{\pi}_j$ is an encoding of $\tilde{\mathbb{i}}[j]$. This completes the proof of decodability because $\mathbf{iD}_{\mathsf{PCP}}$ decodes each $\tilde{\pi}_j$ to $\tilde{\mathbb{i}}[j]$, and these strings together with $\tilde{\mathbb{w}}$ put $\mathbb{x}$ in the multi-indexed relation $\mathcal{R}$.

Let δ_{PCPP} be the PCPP's proximity and δ_{ECC} the code's (relative) distance; recall that $\delta_{\mathsf{PCPP}}\leq\delta_{\mathsf{ECC}}$.

- *Membership*: We claim that there exist strings $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ that place $\mathbb{x}$ in $\mathcal{R}$ and whose encoding has Hamming distance at most δ_{PCPP} from $\tilde{\Pi}_*$; since $\delta_{\mathsf{PCPP}}\leq\delta_{\mathsf{ECC}}$, this implies that $\tilde{\Pi}_*$ decodes to $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}})$. Suppose towards contradiction that there are no such strings. In other words, for every codeword $\widehat{\Pi}_*$ that is close in Hamming distance to $\tilde{\Pi}_*$ we have that $M_{*,\mathbb{x}}(\mathbb{x},\widehat{\Pi}_*)=0$. As a result the PCPP verifier must reject with high probability, which contradicts our assumption that $\mathbf{V}_{\mathsf{PCP}}$ (which runs the PCPP verifier) *accepts* with high probability.

- *Consistency*: We claim that there exist strings $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that their collective encoding is close to $\tilde{\Pi}_*$ and that, for every $j\in[\mathsf{k}]$, $\tilde{\pi}_j$ is close to the encoding of $\tilde{\mathbb{i}}[j]$. As before, since the proximity parameter of the PCPP is smaller than the distance of the code, this implies that $\tilde{\Pi}_*$ decodes to $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}})$ and that $\tilde{\pi}_j$ decodes to $\tilde{\mathbb{i}}[j]$. Suppose towards contradiction that for some $j\in[\mathsf{k}]$ the above condition does not hold: for every $\widehat{\pi}_j$ and $\widehat{\Pi}$ such that $\widehat{\pi}_j$ is close to $\tilde{\pi}_j$ and $\widehat{\Pi}_*$ is close to $\tilde{\Pi}_*$ it holds that $M_j(\widehat{\pi}_j,\widehat{\Pi}_*)=0$. By the soundness of the (*multi-input*) PCPP, the PCPP verifier must reject with high probability, which contradicts our assumption that $\mathbf{V}_{\mathsf{PCP}}$ (which runs the PCPP verifier) *accepts* with high probability.

Complexity measures. The above construction is an index-decodable PCP with polynomial-length proofs and where the verifier makes $O(1)$ queries to each indexer proof and makes $O(1)$ queries to the prover proof. Moreover, the prover proof has alphabet size 2^k since the prover bundles the PCPP consistency test proofs into k-bit symbols; this bundling is possible because the verifier shares randomness between all of the (oblivious) PCPPs in the consistency test. Since the PCPPs are oblivious to the index i, and they share randomness, they all must make the same queries to their oracles. The verifier uses $O(\log |\mathbb{x}|)$ bits of randomness: $O(\log |\mathbb{x}|)$ for the membership test, and $O(\log |\mathbb{x}|)$ for all k consistency tests combined.

3.2.7.2 Achieving constant query complexity over a binary alphabet

We describe how to achieve an index-decodable PCP with constant query complexity per proof over the binary alphabet. The main tool is proof composition. In order to apply proof composition, we define and construct a robust variant of index-decodable PCPs.

Proof composition. Proof composition is a technique to lower the query complexity of PCPs [AS98] and IOPs [BCGRS17]. In proof composition, an "inner" PCP is used to prove that a random execution of the "outer" PCP would have accepted. The inner PCP needs to be a PCPP, which is a PCP system where the verifier has oracle access to its input in addition to the prover's proof, and the soundness guarantee is that if the input is *far* from any input in the language, then the verifier accepts with small probability. To match this, the outer PCP must be *robust*, which means that the soundness guarantee ensures that when the instance is not in the language then not only is a random local view of the verifier rejecting but it is also far (in Hamming distance) from any accepting local view.

Typically the robust outer PCP has small proof length but large query complexity, while the inner PCPP has small query complexity but possibly a large proof length. Composition yields a PCP with small query complexity and small proof length.

We observe that proof composition *preserves decodability* (see Section 3.8.1): if the outer PCP in the composition is index-decodable, then the composed PCP is index-decodable. This is because the composition operation does not change the outer PCP proof and only adds a verification layer to show that the outer verifier accepts.

We thus apply proof composition as follows: the outer PCP is a robust variant of the index-decodable PCP from Section 3.2.7.1; and the inner PCP is a standard PCPP with polynomial proof length. This will complete the proof sketch of Theorem 3.

Defining robust index-decodable PCPs. Our goal is to perform proof composition where the outer PCP is index-decodable. As mentioned above, this requires the PCP to be robust. Our starting point is the index-decodable PCP from Section 3.2.7.1. This PCP does have large query complexity over the binary alphabet ($O(k)$ queries to the prover proof). However, the fact that its queries to the prover proof are already bundled into a constant number of locations over an alphabet of size 2^k implies that we do not have to worry about a "generic" query bundling step and instead only have

to perform a (tailored) robustification step prior to composition. Accordingly, the robustness definition below focuses on the prover proof, and so is the corresponding construction described after.

Definition 3. *A non-adaptive*[7] *index-decodable PCP* $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, (\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}), \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ *for a multi-indexed relation* $\mathcal{R}$ *is* **prover-robustly index-decodable** *with decodability bound* κ_{PCP} *and robustness* σ_{PCP} *if for every* $\mathbb{x}$ *and proofs* $\tilde{\mathbf{\Pi}}_{\mathbb{i}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$ *and* $\tilde{\Pi}$ *if*

$$\Pr_{\alpha}\left[\begin{array}{c|l} \exists\, A', \quad \begin{array}{c}\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x}, \alpha, \tilde{\mathbf{\Pi}}_{\mathbb{i}}[Q_{\mathbb{i}}], A') = 1 \\ \wedge\ \Delta(A', A) \leq \sigma_{\mathsf{PCP}}(|\mathbb{x}|)\end{array} & \begin{array}{l}(Q_{\mathbb{i}}, Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x}, \alpha) \\ A := \{\ \tilde{\Pi}[q] \mid q \in Q_*\ \}\end{array}\end{array}\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)$$

then $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in \mathcal{R}$. *Above* $Q_{\mathbb{i}}$ *and* Q_* *are the queries made to the indexer proofs the prover proof respectively and* $\Delta(A', A)$ *is the relative distance between* A' *and* A.

In other words, if $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \notin \mathcal{R}$ then with high probability not only will the verifier reject but also any set of answers *from the prover proof* that are close in Hamming distance to the real set of answers will also be rejecting.

Robustification. We outline how we transform the index-decodable PCP constructed in Section 3.2.7.1 into a *robust* index-decodable PCP. The techniques follow the robustification step in [BGHSV06b]. The transformation preserves the verifier's randomness complexity $O(\log |\mathbb{x}|)$, which facilitates using this modified PCP as the outer PCP in proof composition.

We apply an error-correcting code separately to each symbol of the prover proof. When the verifier wants to read a symbol from this proof, it reads the codeword encoding the symbol, decodes it, and then continues. It reads the indexer proofs as in the original PCP. This makes the PCP robust because if a few bits of the codeword representing a symbol are corrupted, then it will still be decoded to the same value. The robustness, however, degrades with the number of queries. If the relative distance of the error-correcting code is δ and the original verifier reads q symbols from the prover proof, then the resulting PCP will have robustness $O(\delta/q)$.

Indeed, let $c_1, \ldots, c_q$ be the codewords read by the new PCP verifier from the prover proof, and let $a_1, \ldots, a_q$ be such that a_i is the decoding of c_i. In order to change the decoding into some other set of strings $a'_1, \ldots, a'_q$ that, when received by the verifier, may induce a different decision than $a_1, \ldots, a_q$, it suffices (in the worst case) to change a single codeword to decode to a different value. Since the relative distance of the code is δ, to do this, one must change at least a δ-fraction of the bits of a single codeword, c_i. A δ-fraction of a single codeword is a δ/q-fraction of the whole string of q codewords, $c_1, \ldots, c_q$.

[7] A PCP verifier is non-adaptive if it can be split into two algorithms: $\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}$ chooses which locations to query without accessing its oracles; and $\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}$ receives the results of the queries and decides whether to accept or reject.

In sum, to achieve constant robustness, *we need to begin with an index-decodable PCP with a small number of queries to the prover proof*, but possibly with a large alphabet. It is for this reason that we required this property in Section 3.2.7.1.

3.3 Amplifying round-by-round soundness

In this section we show an amplification lemma for round-by-round soundness of IOPs with decision randomness. Notice that if a k-round IOP has $(\beta_{\mathrm{rbr}}, \delta_{\mathrm{dc}})$-round-by-round soundness, then it has soundness error at most $\mathsf{k} \cdot \beta_{\mathrm{rbr}} + \delta_{\mathrm{dc}}$. Additionally, if a k-round IOP has constant soundness error, then it has $(c^{1/\mathsf{k}}, O(1))$-round-by-round soundness error for some constant $0 < c < 1$.

The following lemma is a generalization of [CCHLRR18], Corollary 5.7, and shows that one can achieve small round-by-round soundness errors for IOPs with relatively little overhead.

Lemma 3.3.1. *Let $m, t \in \mathbb{N}$ be parameters and $\mathcal{R}$ be a relation with a non-adaptive IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with soundness error β_{IOP}. Then $\mathcal{R}$ has an IOP $(\mathbf{P}'_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}})$ with $(\beta_{\mathsf{IOP}}^{m/2\mathsf{k}_{\mathsf{IOP}}}, \beta_{\mathsf{IOP}}^{t/2})$-round-by-round soundness and the following parameters:*

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Rounds	$\mathsf{k}_{\mathsf{IOP}}$
Alphabet size	λ_{IOP}
Proof length	$\mathsf{l}_{\mathsf{IOP}}$
Round queries	$\mathsf{q}_{\mathsf{rnd}}$
Queries per round	$\mathsf{q}_{\mathsf{IOP}}$
Total interaction randomness	$\mathsf{r}_{\mathsf{IOP,int}}$
Decision randomness	$\mathsf{r}_{\mathsf{IOP,dc}}$
Soundness	β_{IOP}
Verifier running time	$\mathsf{vt}_{\mathsf{IOP}}$

$\longrightarrow$

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Rounds	$\mathsf{k}_{\mathsf{IOP}}$
Alphabet size	$\lambda_{\mathsf{IOP}}^{m}$
Proof length	$\mathsf{l}_{\mathsf{IOP}}$
Round queries	$t \cdot \mathsf{q}_{\mathsf{rnd}}$
Queries per round	$t \cdot \mathsf{q}_{\mathsf{IOP}}$
Total interaction randomness	$m \cdot \mathsf{r}_{\mathsf{IOP,int}}$
Decision randomness	$t \cdot \mathsf{r}_{\mathsf{IOP,dc}}$
Round-by-round soundness	$(\beta_{\mathsf{IOP}}^{m/2\mathsf{k}_{\mathsf{IOP}}}, \beta_{\mathsf{IOP}}^{t/2})$
Verifier running time	$m \cdot t \cdot \mathsf{vt}_{\mathsf{IOP}}$

Proof. We augment the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ as follows: first augment the protocol by making m repetitions of the interaction phase and changing the decision phase to check every one of the repetitions simultaneously (at the same locations) and accept if and only if the verifier would have accepted in each execution. This IOP is then augmented by running the decision phase for t different times. Let $(\mathbf{P}'_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}})$ be the final IOP.

Perfect completeness and the efficiency parameters of the IOP follow straightforwardly from the construction.

We turn to showing round-by-round soundness. Fix $\mathbb{x} \notin \mathcal{R}(L)$ and let β_{IOP} be the soundness error of $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$. Then, we claim that for every malicious prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$

the following probability is at most $\beta_{\mathsf{IOP}}^{1/2}$:

$$\Pr_{\alpha_1,\ldots,\alpha_{\mathsf{k_{IOP}}}}\left[\Pr_{\alpha_{\mathsf{dc}}}\left[\mathbf{V}_{\mathsf{IOP}}^{\tilde{\Pi}_1,\ldots,\tilde{\Pi}_{\mathsf{k_{IOP}}},\alpha_1,\ldots,\alpha_{\mathsf{k_{IOP}}}}(\mathbb{x};\alpha_{\mathsf{dc}})=1\right]\geq\beta_{\mathsf{IOP}}^{1/2}\,\middle|\,\begin{array}{r}\tilde{\Pi}_1\leftarrow\tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1)\\ \vdots\\ \tilde{\Pi}_{\mathsf{k_{IOP}}}\leftarrow\tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1,\ldots,\alpha_{\mathsf{k_{IOP}}})\end{array}\right].$$

Indeed, suppose towards contradiction that there exist some $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ such that the probability above is greater than $\beta_{\mathsf{IOP}}^{1/2}$.

In this case, $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ causes $\mathbf{V}_{\mathsf{IOP}}$ to accept at least whenever the verifier chooses interaction randomness for which the internal probability is true (which happens with probability greater than $\beta_{\mathsf{IOP}}^{1/2}$), and where the decision randomness chosen is chosen causes the verifier to accept (which happens with probability $\beta_{\mathsf{IOP}}^{1/2}$). All together, $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ causes the verifier to accept with probability greater than $\beta_{\mathsf{IOP}}^{1/2}\cdot\beta_{\mathsf{IOP}}^{1/2}=\beta_{\mathsf{IOP}}$ in contradiction to the soundness error of the IOP $(\mathbf{P}_{\mathsf{IOP}},\mathbf{V}_{\mathsf{IOP}})$.

Now notice that in order for in order for the verifier $\mathbf{V}'_{\mathsf{IOP}}$ to accept given proofs, it must succeed in all of its t checks of the decision phase. Moreover, these, t checks must succeed for each one of the m independent repetitions. Hence for every $\tilde{\mathbf{P}}'_{\mathsf{IOP}}$, the following probability is at most $\beta_{\mathsf{IOP}}^{m/2}$:

$$\Pr_{\alpha'_1,\ldots,\alpha'_{\mathsf{k_{IOP}}}}\left[\Pr_{\alpha'_{\mathsf{dc}}}\left[\mathbf{V}_{\mathsf{IOP}}^{\prime\tilde{\Pi}'_1,\ldots,\tilde{\Pi}'_{\mathsf{k_{IOP}}},\alpha'_1,\ldots,\alpha'_{\mathsf{k_{IOP}}}}(\mathbb{x};\alpha'_{\mathsf{dc}})=1\right]\geq\beta_{\mathsf{IOP}}^{t/2}\,\middle|\,\begin{array}{r}\tilde{\Pi}'_1\leftarrow\tilde{\mathbf{P}}'_{\mathsf{IOP}}(\alpha'_1)\\ \vdots\\ \tilde{\Pi}'_{\mathsf{k_{IOP}}}\leftarrow\tilde{\mathbf{P}}'_{\mathsf{IOP}}(\alpha'_1,\ldots,\alpha'_{\mathsf{k_{IOP}}})\end{array}\right].$$

We now describe a $(\beta_{\mathsf{IOP}}^{m/2\mathsf{k_{IOP}}},\beta_{\mathsf{IOP}}^{t/2})$-state function state for $(\mathbf{P}'_{\mathsf{IOP}},\mathbf{V}'_{\mathsf{IOP}})$. It is defined as follows for instance $\mathbb{x}$ and transcript tr:

- If $\mathsf{tr}=(\alpha'_1,\tilde{\Pi}'_1,\ldots,\alpha'_{\mathsf{k_{IOP}}},\tilde{\Pi}'_{\mathsf{k_{IOP}}})$ is a full transcript, then $\mathsf{state}(\mathbb{x},\mathsf{tr})=1$ if and only if
$$\Pr_{\alpha'_{\mathsf{dc}}}\left[\,\mathbf{V}_{\mathsf{IOP}}'^{\mathsf{tr}}(\mathbb{x};\alpha'_{\mathsf{dc}})=1\,\right]\geq\beta_{\mathsf{IOP}}^{t/2}\,.$$
- We define $\mathsf{state}(\mathbb{x},\mathsf{tr})$ inductively for transcripts tr that are not full and contain i rounds:
  - If $\mathsf{tr}=(\alpha'_1,\tilde{\Pi}'_1,\ldots,\alpha'_i,\tilde{\Pi}'_i)$ ends in a prover message. Then $\mathsf{state}(\mathbb{x},\mathsf{tr})=1$ if and only if
  $$\Pr_{\alpha'_{i+1}}\left[\,\mathsf{state}(\mathbb{x},\mathsf{tr}||\alpha'_{i+1})=1\,\right]\geq\beta_{\mathsf{IOP}}^{m/2\mathsf{k_{IOP}}}\,.$$
  .
  - If $\mathsf{tr}=(\alpha'_1,\tilde{\Pi}'_1,\ldots,\alpha'_{i-1},\tilde{\Pi}'_{i-1},\alpha'_i)$ ends with a verifier message, then $\mathsf{state}(\mathbb{x},\mathsf{tr})=1$ if and only if there exists a proof $\tilde{\Pi}'_i$ such that $\mathsf{state}(\mathbb{x},\mathsf{tr}||\tilde{\Pi}'_i)=1$.

It follows from the construction that state has decision error $\beta_{\mathsf{IOP}}^{t/2}$. The interaction error of $\beta_{\mathsf{IOP}}^{m/2\mathsf{k_{IOP}}}$ can be shown by a similar argument to that of [CCHLRR18, Proposition 5.5]. □

3.4 Interactive reducibility

We define the notion of interactive reducibility. Then we show basic properties of interactive reducibility: (a) in Section 3.4.1 we show that if a relation has an interactive reduction then it also has an IP; and (b) in Section 3.4.2 we show how to reduce the soundness error of an interactive reduction. Finally, in Section 3.4.3, we show an interactive reduction for any public-coin IP and bounded-output-length interactive reductions for the sumcheck protocol and Shamir's protocol.

Definition 3.4.1. *An* **interactive reduction** *for a relation $\mathcal{R}$ with k_{IR} predicates and soundness error $\varepsilon_{\mathsf{IR}}$ is a ℓ_{IR}-round public-coin protocol $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ (where $\mathbf{V}_{\mathsf{IR}}$ runs in polynomial time) for which there exists a list of predicates $f_0, f_1, \ldots, f_{\mathsf{k}_{\mathsf{IR}}}$ such that the following holds.*

- **Completeness.** *For every $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $j \in [\mathsf{k}_{\mathsf{IR}}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 1$ for every $i \in [\mathsf{t}]$, it holds that*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] = 1\,.$$

- **Soundness.** *For every $\mathbb{x} \notin L(\mathcal{R})$, $j \in [\mathsf{k}_{\mathsf{IR}}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, if there exists $i \in [\mathsf{t}]$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 0$ then for every (computationally unbounded) $\tilde{\mathbf{P}}_{\mathsf{IR}}$ it holds that*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] \leq \varepsilon_{\mathsf{IR}}(\mathbb{x}, \mathsf{t})\,.$$

- **Relation identity.** *$f_0(\mathbb{x}, \mathbb{z}) = 1$ if and only if $\mathbb{x} \in L(\mathcal{R})$.*
- **Triviality.** *$f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z})$ can be computed in time $\mathrm{poly}(|\mathbb{x}|, |\mathbb{z}|)$.*

We call $\mathbb{x}$ the base instance *and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$* round instances*.*

A relation $\mathcal{R}$ is *interactively reducible* with k_{IR} predicates and ℓ_{IR} rounds if $\mathcal{R}$ has an interactive reduction with k_{IR} predicates and k_{IR}-rounds. An interactive reduction has soundness error that is *well-behaved* if $\varepsilon_{\mathsf{IR}}$ is (weakly) monotonically increasing with t and grows at most polynomially with t. All interactive reductions described in this chapter have well-behaved soundness error.

Definition 3.4.2. *An interactive reduction has (polynomially)* **bounded output length** *if there exists $c \in \mathbb{N}$ such that, for every base instance $\mathbb{x}$, witness $\mathbb{w}$, and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, the new instance output by $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ on inputs $(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}})$ has length at most $|\mathbb{x}|^c$.*

3.4.1 Interactive proofs from interactive reductions

We construct an IP for any relation that is interactively reducible.

Theorem 3.4.3. *If a relation $\mathcal{R}$ has an interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ then $\mathcal{R}$ has an IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ with the parameters indicated below.*

Interactive reduction for $\mathcal{R}$	
Number of predicates	k_{IR}
Rounds	ℓ_{IR}
Communication	l_{IR}
Randomness	r_{IR}
Soundness error	$\varepsilon_{\mathsf{IR}}$
Verifier running time	$\mathsf{vt}_{\mathsf{IR}}$
Final predicate time	$\mathsf{ft}_{\mathsf{IR}}$

$\longrightarrow$

IP for $\mathcal{R}$	
Rounds	$\mathsf{k}_{\mathsf{IR}} \cdot \ell_{\mathsf{IR}}$
Communication	$\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{l}_{\mathsf{IR}}$
Randomness	$\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{r}_{\mathsf{IR}}$
Soundness error	$\mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, 1)$
Verifier running time	$\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{vt}_{\mathsf{IR}} + \mathsf{ft}_{\mathsf{IR}}$

Moreover, if $\ell_{\mathsf{IR}} = 1$ then the IP for $\mathcal{R}$ has round-by-round soundness error $\varepsilon_{\mathsf{IR}}(\mathbb{x}, 1)$.

Construction 3.4.4. The IP prover $\mathbf{P}_{\mathsf{IP}}$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the IP verifier $\mathbf{V}_{\mathsf{IP}}$ receives as input the instance $\mathbb{x}$. Letting $\mathbb{z}_0 := \mathbb{x}$, they interact as follows. For $j = 1, \ldots, \mathsf{k}_{\mathsf{IR}}$, $\mathbf{P}_{\mathsf{IP}}$ and $\mathbf{V}_{\mathsf{IP}}$ run the interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ to obtain a new instance

$$\mathbb{z}_j \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_{j-1}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_{j-1}) \rangle \,.$$

Finally, after the interaction, $\mathbf{V}_{\mathsf{IP}}$ checks that $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}_{\mathsf{k}_{\mathsf{IR}}}) = 1$.

Proof of Theorem 3.4.3. We prove completeness, then prove soundness (and round-by-round soundness), and finally analyze complexity measures.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. By completeness of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$, for every $j \in [\mathsf{k}_{\mathsf{IR}}]$, if $f_{j-1}(\mathbb{x}, \mathbb{z}_{j-1}) = 1$, then

$$\Pr\left[f_j(\mathbb{x}, \mathbb{z}_j) = 1 \,\middle|\, \mathbb{z}_j \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_{j-1}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_{j-1}) \rangle \right] = 1 \,.$$

Since $f_0(\mathbb{x}, \mathbb{z}_0) = f_0(\mathbb{x}, \mathbb{x}) = 1$, by induction it holds that $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}_{\mathsf{k}_{\mathsf{IR}}}) = 1$ with probability 1. We conclude that $\Pr[\langle \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathbb{w}), \mathbf{V}_{\mathsf{IP}}(\mathbb{x}) \rangle = 1] = 1$.

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ and a malicious IP prover $\tilde{\mathbf{P}}_{\mathsf{IP}}$ and let $\varepsilon_{\mathsf{IR}} := \varepsilon_{\mathsf{IR}}(\mathbb{x}, 1)$. Let $\tilde{\mathbf{P}}_{\mathsf{IR}}^{(j)}$ be the machine that describes $\tilde{\mathbf{P}}_{\mathsf{IP}}$ during the j-th execution of the interactive reduction. Each $\tilde{\mathbf{P}}_{\mathsf{IR}}^{(j)}$ can be seen as a (potentially malicious) prover for the interactive reduction. We show that $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}_{\mathsf{k}_{\mathsf{IR}}}) = 1$ (a necessary condition for the IP verifier $\mathbf{V}_{\mathsf{IP}}$ to accept) with probability at most $\mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}$.

We show by induction that $f_j(\mathbb{x}, \mathbb{z}_j) = 1$ with probability at most $j \cdot \varepsilon_{\mathsf{IR}}$. For $j = 0$ this holds because $f_0(\mathbb{x}, \mathbb{z}_0) = f_0(\mathbb{x}, \mathbb{x}) = 0$. For $j > 0$, suppose that $f_{j-1}(\mathbb{x}, \mathbb{z}_{j-1}) = 1$ with probability at most $(j-1) \cdot \varepsilon_{\mathsf{IR}}$. It may be that whenever $f_{j-1}(\mathbb{x}, \mathbb{z}_{j-1}) = 1$ it holds holds that $f_j(\mathbb{x}, \mathbb{z}_j) = 1$. However, whenever $f_{j-1}(\mathbb{x}, \mathbb{z}_{j-1}) = 0$, we can invoke the soundness property of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ to deduce that

$$\Pr\left[f_j(\mathbb{x}, \mathbb{z}_j) = 1 \,\middle|\, \mathbb{z}_j \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}^{(j)}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_{j-1}) \rangle \right] \leq \varepsilon_{\mathsf{IR}} \,.$$

By applying the union bound, the probability that $f_j(\mathbb{x}, \mathbb{z}_j) = 1$ is at most $(j-1) \cdot \varepsilon_{\mathsf{IR}} + \varepsilon_{\mathsf{IR}} = j \cdot \varepsilon_{\mathsf{IR}}$.

Round-by-round soundness. Suppose that $\ell_{\mathsf{IR}} = 1$ (that is, $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ has one round) and let $\varepsilon_{\mathsf{IR}} := \varepsilon_{\mathsf{IR}}(\mathbb{x}, 1)$. We focus on the MA case (the prover moves first in the interactive reduction); the AM case can be shown in a similar manner. To show round-by-round soundness we first define a state function state. Let $\mathbb{x}$ be an instance and tr be a transcript of the protocol up to round j (i.e., j full executions of the interactive reduction), and let $(\mathbb{z}_0, \ldots, \mathbb{z}_j)$ be the round instances output by the reduction verifier $\mathbf{V}_{\mathsf{IR}}$ in an execution of the protocol according to tr (if the transcript ends with a prover message, then $\mathbb{z}_j$ is the last round instance specified according to the transcript). Then we set $\mathsf{state}(\mathbb{x}, \mathsf{tr}) := 1$ if and only $f_j(\mathbb{x}, \mathbb{z}_j) = 1$. We show that this is indeed a state function.

- *Empty transcript.* If $\mathsf{tr} = \emptyset$ is the empty transcript then by definition $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ if and only if $f_0(\mathbb{x}, \emptyset) = 0$, which occurs if and only if $\mathbb{x} \notin L(\mathcal{R})$.
- *Prover moves.* Given $\mathbb{x} \notin L(\mathcal{R})$ and a transcript tr. As in the definition of the state function, let $\mathbb{z}_0, \ldots, \mathbb{z}_j$ be the instances generated by $\mathbf{V}_{\mathsf{IR}}$. If $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ then by definition $f_j(\mathbb{x}, \mathbb{z}_j) = 0$. Since no new instance is generated given only the prover's message, $\mathsf{state}(\mathbb{x}, \mathsf{tr}||a) = 0$ for every a.
- *Full transcript.* If tr is a full transcript and $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$, this implies that $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}_{\mathsf{k}_{\mathsf{IR}}}) = 0$. If this is the case then the IP verifier $\mathbf{V}_{\mathsf{IP}}$ rejects.

We argue that $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ has round-by-round soundness $\varepsilon_{\mathsf{IR}}$ with respect to this state function. Fix an instance $\mathbb{x}$, a round number $j \in [\mathsf{k}_{\mathsf{IR}}]$, and a transcript tr that contains messages up to round j (where the next message is a verifier message). Let $(\mathbb{z}_0, \ldots, \mathbb{z}_j)$ be the instances implied by tr. Suppose that $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$. Since the next message is a verifier message, and the interactive reduction is public-coin, we have that the verifier's next message α is a uniformly random string. Thus, we show that

$$\Pr_{\alpha}\left[\, \mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha) = 1 \,\right] \leq \varepsilon_{\mathsf{IR}}\,.$$

Since $\mathsf{state}(\mathsf{tr}) = 0$, we have that $f_j(\mathbb{x}, \mathbb{z}_j) = 0$. The claim, then, is implied by soundness of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$, stating that for any malicious prover $\tilde{\mathbf{P}}_{\mathsf{IR}}$:

$$\Pr\left[\, f_{j+1}(\mathbb{x}, \mathbb{z}_{j+1}) = 1 \,\middle|\, \mathbb{z}_{j+1} \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_j)\rangle \,\right] \leq \varepsilon_{\mathsf{IR}}\,.$$

Complexity measures. We discuss complexity measures of $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$.

- *Round complexity.* The round complexity of the IP is $\mathsf{k}_{\mathsf{IR}} \cdot \ell_{\mathsf{IR}}$ because the IP consists of k_{IR} interactive reductions, each with ℓ_{IR} rounds, ran in sequence.

- *Communication complexity.* The communication complexity of the IP is $\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{l}_{\mathsf{IR}}$, because each of the k_{IR} interactive reductions has communication complexity l_{IR}.

- *Randomness complexity.* The IP verifier uses $\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{r}_{\mathsf{IR}}$ random bits because each of the k_{IR} invocations of the reduction verifier uses r_{IR} random bits.

- *Verifier running time.* The IP verifier runs in time $\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{vt}_{\mathsf{IR}} + \mathsf{ft}_{\mathsf{IR}}$ because it runs k_{IR} invocations of the reduction verifier (whose time complexity is $\mathsf{vt}_{\mathsf{IR}}$) and also runs the predicate $f_{\mathsf{k}_{\mathsf{IR}}}$ (whose time complexity is $\mathsf{ft}_{\mathsf{IR}}$).

□

3.4.2 Error reduction for interactive reducibility

We show that the soundness error of an interactive reduction can be amplified. In order to improve soundness from $\varepsilon_{\mathsf{IR}}$ to $\varepsilon_{\mathsf{IR}}^t$ we have each "new" round instance contain t "original" round instances for the original protocol. Then, when reducing t separate new round instances (each one containing t old round instances), we run the original interactive reduction on all of the old round instances contained within the new round instances.

Theorem 3.4.5. *Let $\mathcal{R}$ be a relation and $t \in \mathbb{N}$ a parameter. If $\mathcal{R}$ has an interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ then $\mathcal{R}$ has an interactive reduction $(\mathbf{P}_{\mathsf{IR}}^{(t)}, \mathbf{V}_{\mathsf{IR}}^{(t)})$ with the parameters indicated below.*

Interactive reduction for $\mathcal{R}$	
Number of predicates	k_{IR}
Rounds	ℓ_{IR}
Output length	s_{IR}
Communication	l_{IR}
Randomness	r_{IR}
Soundness error	$\varepsilon_{\mathsf{IR}}$
Verifier running time	$\mathsf{vt}_{\mathsf{IR}}$
Final predicate time	$\mathsf{ft}_{\mathsf{IR}}$

$\longrightarrow$

Interactive reduction for $\mathcal{R}$ with reduced error	
Number of predicates	k_{IR}
Rounds	$\ell_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$
Output length	$t \cdot \mathsf{s}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$
Communication	$t \cdot \mathsf{l}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$
Randomness	$t \cdot \mathsf{r}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$
Soundness error	$\varepsilon_{\mathsf{IR}}^t(\mathbb{x}, t \cdot \mathsf{t})$
Verifier running time	$t \cdot \mathsf{vt}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$
Final predicate time	$t \cdot \mathsf{ft}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$

Construction 3.4.6. We define the new predicates and then describe the new interactive reduction.

The predicates. Let f_j be the j-th predicate in $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$. Define the new predicate $f_j^{(t)}$ for $(\mathbf{P}_{\mathsf{IR}}^{(t)}, \mathbf{V}_{\mathsf{IR}}^{(t)})$ so that $f_j^{(t)}(\mathbb{x}, \mathbb{z}) = 1$ if and only if $\mathbb{z} = (\mathbb{z}_1, \ldots, \mathbb{z}_t)$ and $f_j(\mathbb{x}, \mathbb{z}_m) = 1$ for all $m \in [t]$.

The reduction. The reduction prover $\mathbf{P}_{\mathsf{IR}}^{(t)}$ and reduction verifier $\mathbf{V}_{\mathsf{IR}}^{(t)}$ receive as input a base instance $\mathbb{x}$ and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ where $\mathbb{z}_i := (\mathbb{z}_{i,1}, \ldots, \mathbb{z}_{i,t})$. They interact as follows.

- They run the interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ with base instance $\mathbb{x}$ and round instances $(\mathbb{z}_{1,1}, \ldots, \mathbb{z}_{1,t}, \ldots, \mathbb{z}_{\mathsf{t},1}, \ldots, \mathbb{z}_{\mathsf{t},t})$ for t times in parallel with fresh randomness in each execution.
- Output $\mathbb{z}' := (\mathbb{z}'_1, \ldots, \mathbb{z}'_t)$, where $\mathbb{z}'_i$ is the result of the i-th independent execution of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$.

Proof of Theorem 3.4.5. First we argue completeness, then soundness of the interactive reduction and triviality of the predicate $f_{\mathsf{k}_{\mathsf{IR}}}$, and finally analyze the other complexity measures of the protocol.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ with $f_j(\mathbb{x}, \mathbb{z}_1) = \ldots = f_j(\mathbb{x}, \mathbb{z}_{\mathsf{t}}) = 1$. Completeness of the interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ implies that with probability 1 every output $\mathbb{z}'_i$ satisfies $f_{j+1}(\mathbb{x}, \mathbb{z}'_i) = 1$; in turn, by definition of the new predicates, this means that $\mathbb{z}' := (\mathbb{z}'_1 \ldots, \mathbb{z}'_t)$ is such that $f_j^{(t)}(\mathbb{x}, \mathbb{z}') = 1$.

Soundness. Fix $\mathbb{x}$ and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ where $f_j^{(t)}(\mathbb{x}, \mathbb{z}_i) = 0$ for some $i \in [\mathsf{t}]$. This implies that there exists $\mathbb{z}_{i,m} \in \mathbb{z}_i$ such that $f_j^{(t)}(\mathbb{x}, \mathbb{z}_{i,m}) = 0$. This is one of the instances in the list $(\mathbb{z}_{1,1}, \ldots, \mathbb{z}_{1,t}, \ldots, \mathbb{z}_{\mathsf{t},1}, \ldots, \mathbb{z}_{\mathsf{t},t})$ given as input to the interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$. Thus, for each execution in the protocol we have, by the definition of reducible soundness that for every $\tilde{\mathbf{P}}_{\mathsf{IR}}$:

$$\Pr\left[\, f_{j+1}(\mathbb{x}, \mathbb{z}) = 1 \,\middle|\, \mathbb{z} \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_{1,1}, \ldots, \mathbb{z}_{1,t}, \ldots, \mathbb{z}_{\mathsf{t},1}, \ldots, \mathbb{z}_{\mathsf{t},t}) \rangle \,\right] \leq \varepsilon_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})\,.$$

Since each of the t executions is independent, with probability at least $1 - \varepsilon_{\mathsf{IR}}^t(\mathbb{x}, t \cdot \mathsf{t})$ there exists $i \in [t]$ with $f_{j+1}(\mathbb{x}, \mathbb{z}'_i) = 0$. This implies that $f_{j+1}^{(t)}(\mathbb{x}, \mathbb{z}') = 0$ with probability at least $1 - \varepsilon_{\mathsf{IR}}^t(\mathbb{x}, t \cdot \mathsf{t})$.

Triviality. Triviality of $f_{\mathsf{k}_{\mathsf{IR}}}^{(t)}$ follows directly from the triviality of $f_{\mathsf{k}_{\mathsf{IR}}}$.

Complexity measures. We discuss complexity measures of the new interactive reduction $(\mathbf{P}_{\mathsf{IR}}^{(t)}, \mathbf{V}_{\mathsf{IR}}^{(t)})$. Running the new interactive reduction with t round instances is essentially t executions of the interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ with $t \cdot \mathsf{t}$ round instances each. Thus all complexity measures depend on the complexity of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ with $t \cdot \mathsf{t}$ round instances.

- *Output length.* The output of the protocol has length $t \cdot \mathsf{s}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$.
- *Round complexity.* The protocol has $\ell_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$ rounds.
- *Communication complexity.* The communication complexity of the protocol is $t \cdot \mathsf{l}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$.
- *Randomness complexity.* The verifier of the new interactive reduction uses $t \cdot \mathsf{r}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$ random bits.
- *Verifier running time.* The verifier of the new interactive reduction runs in time $t \cdot \mathsf{vt}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$.
- *Final predicate time.* The final predicate of the new interactive reduction runs in time $t \cdot \mathsf{ft}_{\mathsf{IR}}(\mathbb{x}, t \cdot \mathsf{t})$.

□

3.4.3 Relations with interactive reducibility

We present interactive reductions for various relations. In Section 3.4.3.1 we show that every language with an IP has an interactive reduction, albeit not one with bounded output length. Then we show how to achieve bounded output length for specific classes of relations: 1. the sumcheck protocol in Section 3.4.3.2; and 2. Shamir's protocol in Section 3.4.3.3;

3.4.3.1 Interactive reductions for any IP

We show that any IP can be transformed into an IP with interactive reducibility where the base instance is the instance for the IP, and the round instances are sets of partial transcripts. The length of a merged round instance grows with the number of partial transcripts in all of the reduction instances combined, and so this reduction does not have bounded output-length.

Theorem 3.4.7. *If a relation $\mathcal{R}$ has a k_{IP}-round public-coin IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ then $\mathcal{R}$ has an interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ with the parameters below for every $m > 0$ that divides k_{IP}.*

IP for $\mathcal{R}$	
Rounds	k_{IP}
Per-round communication	l_{IP}
Per-round randomness	r_{IP}
Round-by-round soundness error	β_{rbr}
Per-round verifier running time	vt
Decision time	d_{IP}

$\longrightarrow$

Interactive reduction for $\mathcal{R}$ for round j given $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$	
Number of predicates	$\mathsf{k}_{\mathsf{IP}}/m$
Rounds	m
Output length	$\mathsf{t} \cdot m \cdot \mathsf{l}_{\mathsf{IP}} + \sum_{i \in [\mathsf{t}]} \lvert\mathbb{z}_i\rvert$
Communication	$\mathsf{t} \cdot m \cdot \mathsf{l}_{\mathsf{IP}}$
Randomness	$m \cdot \mathsf{r}_{\mathsf{IP}}$
Soundness error	$m \cdot \beta_{\mathrm{rbr}}$
Verifier running time	$\mathsf{t} \cdot m \cdot \mathsf{vt}_{\mathsf{IP}}$
Final predicate time	d_{IP}

Construction 3.4.8. Let state be the state function of $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$. We define the predicates and then describe the interactive reduction for the relation described by this IP.

The predicates. For each $j \in [\mathsf{k}_{\mathsf{IP}}/m]$, we define $f_j(\mathbb{x}, \mathbb{z}) := 1$ if and only if the following are true:
1. $\mathbb{z}$ is a set of $(j \cdot m)$-round partial transcript ending with a prover message.
2. For every $\mathsf{tr} \in \mathbb{z}$, $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 1$.

3. If $\mathbb{x} \in L(\mathcal{R})$ and $j \neq \mathsf{k}_{\mathsf{IP}}/m$, then we require that very $\mathsf{tr} \in \mathbb{z}$ is consistent with the honest prover: Suppose $\mathsf{tr} = (a_1, \alpha_1, \ldots, a_{j \cdot m})$ then $a_1 := \mathbf{P}_{\mathsf{IP}}(\mathbb{x})$ and for every $1 < k \leq j \cdot m$, $a_k := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, a_1, \alpha_1, \ldots, a_{k-1}, \alpha_{k-1})$.

The reduction. The reduction prover $\mathbf{P}_{\mathsf{IR}}$ and reduction verifier $\mathbf{V}_{\mathsf{IR}}$ receive as input a base instance $\mathbb{x}$ and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ where each $\mathbb{z}_i$ is a set of j-round partial transcripts ending in a prover message. The reduction prover $\mathbf{P}_{\mathsf{IR}}$ additionally receives as input a witness $\mathbb{w}$. The protocol is as follows:

1. For $k = 1$ to m:
   (a) $\mathbf{V}_{\mathsf{IR}}$: Send a uniformly random message α_k corresponding to the IP verifier's $(j \cdot m + k)$-th message.
   (b) $\mathbf{P}_{\mathsf{IR}}$: For every $\mathsf{tr} \in \bigcup_{i \in [\mathsf{t}]} \mathbb{z}_i$, send the prover message $a_{\mathsf{tr},k} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathbb{w}, \mathsf{tr}, \alpha_1, \ldots, \alpha_k)$.
2. $\mathbf{V}_{\mathsf{IR}}$: Output $\mathbb{z}' := \{\mathsf{tr}||\alpha_1||a_{\mathsf{tr},1}|| \ldots ||\alpha_m||a_{\mathsf{tr},m}\}_{\mathsf{tr} \in \bigcup_{i \in [\mathsf{t}]} \mathbb{z}_i}$.

Proof of Theorem 3.4.7. First we argue completeness, then soundness and then show triviality of the predicate $f_{\mathsf{k}_{\mathsf{IP}}/m}$.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $j \in [\mathsf{k}_{\mathsf{IP}}/m]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_1) = \cdots = f_{j-1}(\mathbb{x}, \mathbb{z}_{\mathsf{t}}) = 1$. This implies that $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 1$ for every $\mathsf{tr} \in \bigcup_{i \in [\mathsf{t}]} \mathbb{z}_i$ and that every transcript is consistent with the honest prover. We now show that for every such transcript tr and $\alpha_1, \ldots, \alpha_m$, setting $a_{\mathsf{tr},k} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathbb{w}, \mathsf{tr}, \alpha_1, a_{\mathsf{tr},1}, \ldots, a_{\mathsf{tr},k-1}, \alpha_k)$, we have

$$\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha_1||a_{\mathsf{tr},1}|| \ldots ||\alpha_m||a_{\mathsf{tr},m}) = 1\,.$$

This implies that $f_{j+1}(\mathbb{x}, \mathbb{z}') = 1$, since $f_{j+1}(\mathbb{x}, \mathbb{z}') = 1$ if and only if for every tr the state of the resulting transcript after running the protocol is 1.

Notice that any for any transcript tr that is consistent with the honest prover and any α: $\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha||a_{\mathsf{tr}}) = 1$ where $a_{\mathsf{tr}} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}||\alpha)$. Suppose towards contradiction of perfect completeness of the underlying IP that this was not the case. Then for some choice of α the honest prover sends a message that sets the state to 0. Since the protocol has non-zero round-by-round soundness error, once the state is set to zero, there is a non-zero probability that the final transcript has state 0, which will imply that the verifier rejects. Since tr was consistent with the honest prover, there is a non-zero probability that it is generated in an honest execution of the protocol. Therefore, there is non-zero probability that the honest prover will not convince the verifier in a correct execution, in contradiction to perfect completeness of the IP.

The above argument shows that for every transcript, its state will remain 1. By construction, every new transcript $(\mathsf{tr}||\alpha_1||a_{\mathsf{tr},1}|| \ldots ||\alpha_m||a_{\mathsf{tr},m})$ is consistent with the honest prover. We therefore conclude that $f_j(\mathbb{x}, \mathbb{z}') = 1$.

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$, $j \in [\mathsf{k}_{\mathsf{IP}}/m]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ and a (computationally unbounded) malicious reduction prover $\tilde{\mathbf{P}}_{\mathsf{IR}}$. Before we begin our analysis, recall that round-by-round soundness of the IP, for any transcript tr with $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$:

$$\Pr_{\alpha}[\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha) = 1] \leq \beta_{\mathrm{rbr}}\,.$$

Moreover, by the properties of the state function, since $\mathbb{x} \notin L(\mathcal{R})$, if α is such that $\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha) = 0$ then for any a we have that $\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha_j||a) = 0$. Using induction, we can conclude that for any tr with $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$:

$$\Pr_{\alpha_1,\ldots,\alpha_m}\left[\mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha_1||a_{\mathsf{tr},1}||\ldots||\alpha_m||a_{\mathsf{tr},m}) = 1 \;\middle|\; \begin{array}{r} a_{\mathsf{tr},1} := \tilde{\mathbf{P}}_{\mathsf{IR}}(\mathsf{tr}, \alpha_1) \\ \vdots \\ a_{\mathsf{tr},m} := \tilde{\mathbf{P}}_{\mathsf{IR}}(\mathsf{tr}, \alpha_1, a_{\mathsf{tr},1}, \ldots, a_{\mathsf{tr},m-1}, \alpha_m) \end{array}\right] \leq m \cdot \beta_{\mathrm{rbr}}\,. \tag{3.2}$$

Suppose that there exists $i \in [\mathsf{t}]$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 0$. This implies that there exists a transcript $\mathsf{tr} \in \bigcup_{i\in[\mathsf{t}]} \mathbb{z}_i$ with $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$ (notice that the requirement of consistency with the honest prover described in Item 3 is only for $\mathbb{x} \in L(\mathcal{R})$). By Equation 3.2 and since $(\mathsf{tr}||\alpha_1||a_{\mathsf{tr},1}||\ldots||\alpha_m||a_{\mathsf{tr},m})$ is added to the new instance $\mathbb{z}'$, this implies that with probability at least $1 - m \cdot \beta_{\mathrm{rbr}}$, $f_j(\mathbb{x}, \mathbb{z}') = 0$ as required.

Triviality. $f_{\mathsf{k_{IP}}/m}(\mathbb{x}, \mathbb{z}) = 1$ if and only if all of the transcripts in $\mathbb{z}$ are complete transcripts and have state 1 (notice that since this is the last round, we do not have the requirement of consistency with the honest prover described in Item 3). This can be verified by running $\mathbf{V}_{\mathsf{IP}}$ on every $\mathsf{tr} \in \mathbb{z}$, since $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}) = \mathsf{state}(\mathbb{x}, \mathsf{tr})$.

Complexity measures. We discuss complexity measures of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$.

- *Output length.* The new instance size is $\mathsf{t} \cdot m \cdot \mathsf{l}_{\mathsf{IP}} + \sum_{i\in[\mathsf{t}]} |\mathbb{z}_i|$.
- *Round complexity.* The protocol has m rounds.
- *Communication complexity.* The communication complexity of the protocol is $\mathsf{t} \cdot m \cdot \mathsf{l}_{\mathsf{IP}}$ where l_{IP} is the per-round communication complexity of $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{IR}}$ uses r_{IP} random bits where $m \cdot \mathsf{r}_{\mathsf{IP}}$ is the per-round randomness complexity of $\mathbf{V}_{\mathsf{IP}}$.
- *Verifier running time.* $\mathbf{V}_{\mathsf{IR}}$ runs the IP verifier for $\mathsf{t} \cdot m$ times in total time $\mathsf{t} \cdot m \cdot \mathsf{vt}_{\mathsf{IP}}$.
- *Final predicate time.* The final predicate can be checked in time d_{IP}.

□

3.4.3.2 Sumcheck language

Definition 3.4.9. *A* **sumcheck instance** *has the form* $(\mathbb{F}, H, n, d, p, \gamma)$ *where* $\mathbb{F}$ *is a field,* $H \subseteq \mathbb{F}$, $n \in \mathbb{N}$ *is a number of variables,* $d \in \mathbb{N}$ *is a degree bound,* $p\colon \mathbb{F}^n \to \mathbb{F}$ *is an* n*-variate polynomial of individual degree at most* d *(given to the verifier as an oracle) and* $\gamma \in \mathbb{F}$ *is a claimed sum. We define* $(\mathbb{F}, H, n, d, p, \gamma) \in L_\Sigma$ *if and only if*

$$\sum_{\alpha_1,\ldots,\alpha_n \in H} p(\alpha_1, \ldots, \alpha_n) = \gamma\,.$$

Theorem 3.4.10. *The sumcheck language L_Σ has an interactive reduction that, for instances $\mathbb{x} = (\mathbb{F}, H, n, d, p, \gamma)$, has the following parameters.*

Interactive reduction for L_Σ	
Number of predicates	n
Messages	2
Output length	$\lvert\mathbb{x}\rvert$
Communication	$O(n\mathtt{t}d^2 \log \lvert\mathbb{F}\rvert)$
Randomness	$2 \log \lvert\mathbb{F}\rvert$
Soundness error	$O\left(n\mathtt{t}d/\lvert\mathbb{F}\rvert\right)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathtt{t})$
Final predicate time	1 call to p

Construction 3.4.11. We define the predicates and then describe the interactive reduction.

The predicates. For a base instance $\mathbb{x} = (\mathbb{F}, H, n, d, p, \gamma)$, for each $j \in [n]$, we define $f_j(\mathbb{x}, \mathbb{z}) = 1$ if and only if, parsing $\mathbb{z} = (r_1, \ldots, r_j, \gamma_j) \in \mathbb{F}^{j+1}$, we have

$$\sum_{\alpha_{j+1},\ldots,\alpha_n \in H} p(r_1, \ldots, r_j, \alpha_{j+1}, \ldots, \alpha_n) = \gamma_j \,.$$

The reduction. The reduction prover $\mathbf{P}_{\mathsf{IR}}$ and reduction verifier $\mathbf{V}_{\mathsf{IR}}$ receive as input a base instance $\mathbb{x}$ and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_\mathtt{t}$. Parse each $\mathbb{z}_i := (r_1^{(i)}, \ldots, r_j^{(i)}, \gamma_j^{(i)})$. Let $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}} \colon \mathbb{F} \to \mathbb{F}^j$ be the polynomial of degree less than t such that $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(i) = (r_1^{(i)}, \ldots, r_j^{(i)})$ for every $i \in [\mathtt{t}]$;[8] for any $\gamma \in \mathbb{F}$, $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(\gamma)$ can be computed in $\mathrm{poly}(\mathtt{t})$ operations. The interactive reduction is as follows:

- $\mathbf{P}_{\mathsf{IR}}$: Send the polynomial $g \in \mathbb{F}[X_1, X_2]$ defined as:

$$g(X_1, X_2) := \sum_{\alpha_{j+2},\ldots,\alpha_n \in H} p(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(X_1), X_2, \alpha_{j+2}, \ldots, \alpha_n) \,. \tag{3.3}$$

- $\mathbf{V}_{\mathsf{IR}}$: Receive a bivariate polynomial $\tilde{g} \in \mathbb{F}[X_1, X_2]$ of degree at most $j \cdot d \cdot (\mathtt{t} - 1)$ in X_1 and degree at most d in X_2.
  1. *Consistency:* Check that for every $i \in [\mathtt{t}]$ it holds that $\sum_{\alpha \in H} \tilde{g}(i, \alpha) = \gamma_j^{(i)}$. (Reject if not.)
  2. *Generate new instance:*
     (a) Sample uniformly random field elements $\alpha, r^* \leftarrow \mathbb{F}$ and send them to $\mathbf{P}_{\mathsf{IR}}$.
     (b) Output the new instance $\mathbb{z}' := (r'_1, \ldots, r'_j, r^*, \gamma_{j+1})$ where $(r'_1, \ldots, r'_j) := I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(\alpha)$ and $\gamma_{j+1} := \tilde{g}(\alpha, r^*)$.

[8] Here we implicitly associate the set $[\mathtt{t}]$ with an arbitrary set $S \subseteq \mathbb{F}$ of size t.

Proof of Theorem 3.4.10. We argue completeness, then soundness, and then triviality of the predicate f_n.

Completeness. Fix $\mathbb{x} = (\mathbb{F}, H, n, d, p, \gamma) \in L_\Sigma$ and $\mathbb{z}_1, \dots, \mathbb{z}_t$ such that $f_j(\mathbb{x}, \mathbb{z}_1) = \cdots = f_j(\mathbb{x}, \mathbb{z}_t) = 1$ where $\mathbb{z}_i := (r_1^{(i)}, \dots, r_j^{(i)}, \gamma_j^{(i)})$. We argue that

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \dots, \mathbb{z}_t), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \dots, \mathbb{z}_t)\rangle \,\right] = 1\,.$$

Let g be defined as in Equation 3.3. Notice that for every $i \in [t]$,

$$g(i, X_2) = \sum_{\alpha_{j+2}, \dots, \alpha_n \in H} p(r_1^{(i)}, \dots, r_j^{(i)}, X_2, \alpha_{j+2}, \dots, \alpha_n)\,,$$

which, since $f_j(\mathbb{x}, \mathbb{z}_i) = 1$, implies that $\sum_{\alpha \in H} g(i, \alpha) = \gamma_j^{(i)}$. Therefore the reduction verifier $\mathbf{V}_{\mathsf{IR}}$ does not reject in the consistency test (in Item 1). Moreover, by the definition of g, for every α and r^*,

$$\begin{aligned}\sum_{\alpha_{j+2}, \dots, \alpha_n \in H} p(r'_1, \dots, r'_j, r^*, \alpha_{j+2}, \dots, \alpha_n) &= \sum_{\alpha_{j+2}, \dots, \alpha_n \in H} p(I_{\mathbb{z}_1, \dots, \mathbb{z}_t}(\alpha), r^*, \alpha_{j+2}, \dots, \alpha_n) \\ &= g(\alpha, r^*)\,.\end{aligned}$$

Therefore we have $f_i(\mathbb{x}, \mathbb{z}') = 1$ with probability 1.

Soundness. Fix $\mathbb{x} = (\mathbb{F}, H, n, d, p, \gamma) \notin L_\Sigma$ and $\mathbb{z}_1, \dots, \mathbb{z}_t$ where $\mathbb{z}_i := (r_1^{(i)}, \dots, r_j^{(i)}, \gamma_j^{(i)})$. Suppose that $f_j(\mathbb{x}, \mathbb{z}_i) = 0$ for some $i \in [t]$. Fix a malicious reduction prover $\tilde{\mathbf{P}}_{\mathsf{IR}}$. We show that

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \dots, \mathbb{z}_t)\rangle \,\right] \le \frac{d \cdot (1 + n \cdot (t-1))}{|\mathbb{F}|}\,.$$

If $\tilde{g}$ does not pass the consistency test (in Item 1) then the reduction verifier outputs $\perp$, and we have $f_{j+1}(\mathbb{x}, \perp) = 0$. Thus, we can assume that $\tilde{g}$ passes the consistency test (that is, for every $i \in [t]$ it holds that $\sum_{\alpha \in H} \tilde{g}(i, \alpha) = \gamma_j^{(i)}$). Let g be defined as in Equation 3.3 with respect to $\mathbb{x}$ and $\mathbb{z}_1, \dots, \mathbb{z}_t$.

We have that $g \not\equiv \tilde{g}$ due to the fact that $f_j(\mathbb{x}, \mathbb{z}_i) = 0$, which implies that

$$\sum_{\alpha \in H} g(i, \alpha) = \sum_{\alpha_{j+1}, \dots, \alpha_n \in H} p(r_1^{(i)}, \dots, r_j^{(i)}, \alpha_{j+1}, \dots, \alpha_n) \neq \gamma_j^{(i)} = \sum_{\alpha \in H} \tilde{g}(i, \alpha)\,,$$

Thus, by applying the Schwartz–Zippel lemma, and recalling that g and $\tilde{g}$ both have degree $j \cdot d \cdot (t-1)$ in their first variable and d in their second variable, we get:

$$\Pr_{\alpha, r^*}\left[\, g(\alpha, r^*) = \tilde{g}(\alpha, r^*) \,\right] \le \frac{d \cdot (1 + j \cdot (t-1))}{|\mathbb{F}|} \le \frac{d \cdot (1 + n \cdot (t-1))}{|\mathbb{F}|}\,.$$

For any choice of α and r^* such that $g(\alpha, r^*) \neq \tilde{g}(\alpha, r^*)$, we have that

$$\sum_{\alpha_{j+2},\ldots,\alpha_n \in H} p(r'_1, \ldots, r'_j, r^*, \alpha_{j+2}, \ldots, \alpha_n) = g(\alpha, r^*) \neq \tilde{g}(\alpha, r^*) = \gamma_{j+1}\,,$$

and so $f_{j+1}(\mathbb{x}, (r'_1, \ldots, r'_j, r^*, \gamma_{j+1})) = 0$ with probability at least $1 - \frac{d\cdot(1+n\cdot(\mathrm{t}-1))}{|\mathbb{F}|}$.

Triviality. By definition, $f_n(\mathbb{x}, \mathbb{z}) = f_n((\mathbb{F}, H, n, d, p, \gamma), (r_1, \ldots, r_n, \gamma_n)) = 1$ if and only if $p(r_1, \ldots, r_n) = \gamma_n$, which can be efficiently verified by querying p once.

Complexity measures. We discuss complexity measures of the protocol $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$.

- *Output length.* The output is a list of $j + 2 \leq n + 2$ field elements. Thus, the maximal length output is at most $O(n \cdot \log|\mathbb{F}|)$, which is a fixed polynomial in the input length.
- *Rounds.* The interactive reduction has one round.
- *Communication complexity.* The prover sends g which is bivariate and has degree at most $n \cdot d \cdot (\mathrm{t} - 1)$ in its first variable and d in its second variable. This requires sending $O(n\mathrm{t}d^2)$ field elements. The verifier sends two field elements. Thus the communication complexity of the protocol is $O(n\mathrm{t}d^2 \log|\mathbb{F}|)$ bits.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{IR}}$ uses $2\log|\mathbb{F}|$ random bits.
- *Verifier running time.* $\mathbf{V}_{\mathsf{IR}}$ runs in time $\mathrm{poly}(n, \log|\mathbb{F}|) = \mathrm{poly}(|\mathbb{x}|)$.
- *Final predicate time.* Computing the final predicate requires one call to p.

□

3.4.3.3 Shamir's protocol

We show an interactive reduction for a class of languages that contains all of PSPACE. For notational convenience, we define an operator that acts either as a sum or product, depending on the index of the variable it is applied over. For every $j \in \mathbb{N}$ let:

$$\mathop{\Psi}_{\alpha_j \in H} := \begin{cases} \sum_{\alpha_j \in H} & j \text{ is odd} \\ \prod_{\alpha_j \in H} & j \text{ is even} \end{cases}.$$

Given polynomials $p_0, \ldots, p_n$ where each p_j is a j-variate polynomial over a field $\mathbb{F}$, we derive polynomials $h_0, \ldots, h_n$ where $h_j \colon \mathbb{F}^j \to \mathbb{F}$ is defined recursively as follows:

- $h_n(X_1, \ldots, X_n) := p_n(X_1, \ldots, X_n)$.
- $h_j(X_1, \ldots, X_j) := p_j(X_1, \ldots, X_j) \cdot \mathbf{\Psi}_{\alpha_{j+1}} \Big(h_{j+1}(X_1, \ldots, X_j, \alpha_{j+1})\Big)$.

Observe that using this notation:

$$h_0 = p_0 \cdot \sum_{\alpha_1 \in H} \left[p_1(\alpha_1) \cdot \prod_{\alpha_2 \in H} \left[p_2(\alpha_1, \alpha_2) \cdot \sum_{\alpha_3 \in H} \left[\right.\right.\right.$$
$$\left.\left.\left. \dots \left[p_{n-1}(\alpha_1, \dots, \alpha_{n-1}) \cdot \Psi_{\alpha_n \in H} p_n(\alpha_1, \dots, \alpha_n) \right] \dots \right] \right] \right].$$

Definition 3.4.12. *A* **Shamir instance** *has the form* $(\mathbb{F}, H, n, d, p_0, \dots, p_n, \gamma)$ *where* $\mathbb{F}$ *is a field,* $H \subseteq \mathbb{F}$*,* $n \in \mathbb{N}$ *is a number of variables,* $d \in \mathbb{N}$ *is a degree bound, each* $p_j \colon \mathbb{F}^j \to \mathbb{F}$ *is an* j*-variate polynomial where the derived polynomials* $h_0, \dots, h_n$ *have individual degree at most* d *and* $\gamma \in \mathbb{F}$ *is a claimed value. We define* $(\mathbb{F}, H, n, d, p_0, \dots, p_n, \gamma) \in L_{\mathsf{Shmr}}$ *if and only if* $h_0 = \gamma$*.*

Note that a sumcheck instance (as defined in Section 3.4.3.2) is a Shamir instance where $p_0, \dots, p_{n-1}$ are equal to the constant 1 polynomial, and $p_n(X_1, \dots, X_n) \triangleq p'_n(X_1, X_3, \dots, X_{n-1})$ (assuming n is even) for some polynomial p'_n of individual degree at most d.

Theorem 3.4.13. *The Shamir language* L_{Shmr} *has an interactive reduction that, for instances* $\mathbb{x} = (\mathbb{F}, H, n, d, p_0, \dots, p_n, \gamma)$*, has the following parameters.*

Interactive reduction for L_{Shmr}	
Number of predicates	n
Messages	2
Output length	$\lvert\mathbb{x}\rvert$
Communication	$O(ntd^2 \log \lvert\mathbb{F}\rvert)$
Randomness	$2 \log \lvert\mathbb{F}\rvert$
Soundness error	$O\left(ntd/\lvert\mathbb{F}\rvert\right)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathtt{t})$
Final predicate time	1 call to p_n

Shamir [Sha92] showed how deciding every language in PSPACE can be reduced to computing the value of an arithmetic expression of the form of Shamir instances. Specifically, Shamir's reduction takes a TQBF formula and transforms it into a "simple" TQBF formula, roughly defined as one in which every occurrence of every variable is separated from its quantification point by at most one universal quantifier (and arbitrarily many other symbols). This ensures that when arithmetizing the formula, one ends up with an expression in which the degree of every variable is polynomial.

Using the above theorem, and Shamir's reduction we get the following corollary showing that every language in PSPACE has an interactive reduction with bounded output length albeit with polynomially-many predicates.

Corollary 3.4.14. *Every* $L \in$ PSPACE *has an interactive reduction that, for instances* $\mathbb{x}$*, has the following parameters:*

Interactive reduction for L	
Number of predicates	$\text{poly}(\lvert\mathbb{x}\rvert)$
Messages	2
Output length	$\text{poly}(\lvert\mathbb{x}\rvert)$
Communication	$\text{poly}(\lvert\mathbb{x}\rvert)$
Randomness	$\text{poly}(\lvert\mathbb{x}\rvert)$
Soundness error	$O\left(\mathtt{t}/2^{\lvert\mathbb{x}\rvert}\right)$
Verifier running time	$\text{poly}(\lvert\mathbb{x}\rvert, \mathtt{t})$
Final predicate time	$\text{poly}(\lvert\mathbb{x}\rvert)$

Construction 3.4.15. We define the predicates and then describe the interactive reduction.

The predicates. For a base instance $\mathbb{x} = (\mathbb{F}, H, n, d, p_0, \ldots, p_n, \gamma)$, for each $j \in [n]$, we define $f_j(\mathbb{x}, \mathbb{z}) = 1$ if and only if, parsing $\mathbb{z} = (r_1, \ldots, r_j, \gamma_j) \in \mathbb{F}^j$, we have $h_j(r_1, \ldots, r_j) = \gamma_j$.

The reduction. The reduction prover $\mathbf{P}_{\mathsf{IR}}$ and reduction verifier $\mathbf{V}_{\mathsf{IR}}$ receive as input a base instance $\mathbb{x}$ and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_\mathtt{t}$. Parse each $\mathbb{z}_i := (r_1^{(i)}, \ldots, r_j^{(i)}, \gamma_j^{(i)})$. Let $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}} \colon \mathbb{F} \to \mathbb{F}^j$ be the polynomial of degree less than $\mathtt{t}$ such that $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(i) = (r_1^{(i)}, \ldots, r_j^{(i)})$ for every $i \in [\mathtt{t}]$ (we implicitly associate the set $[\mathtt{t}]$ with an arbitrary set $S \subseteq \mathbb{F}$ of size $\mathtt{t}$.); for any $\gamma \in \mathbb{F}$, $I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(\gamma)$ can be computed in $\text{poly}(\mathtt{t})$ operations. The interactive reduction is as follows:

- $\mathbf{P}_{\mathsf{IR}}$: Send the polynomial $g \in \mathbb{F}[X_1, X_2]$ defined as:

$$g(X_1, X_2) := h_{j+1}(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(X_1), X_2)\,. \tag{3.4}$$

- $\mathbf{V}_{\mathsf{IR}}$: Receive a bivariate polynomial $\tilde{g} \in \mathbb{F}[X_1, X_2]$ of degree at most $j \cdot d \cdot (\mathtt{t} - 1)$ in X_1 and degree at most d in X_2.
  1. *Consistency:* Check that for every $i \in [\mathtt{t}]$ it holds that

$$p_j(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(i)) \cdot \sum_{\alpha_{j+1} \in H} \tilde{g}(i, \alpha_{j+1}) = \gamma_j^{(i)}\,.$$

     (Reject if not.)
  2. *Generate new instance:*
     (a) Sample uniformly random field elements $\alpha, r^* \leftarrow \mathbb{F}$ and send them to $\mathbf{P}_{\mathsf{IR}}$.
     (b) Output the new instance $\mathbb{z}' := (r'_1, \ldots, r'_j, r^*, \gamma_{j+1})$ where $(r'_1, \ldots, r'_j) := I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathtt{t}}(\alpha)$ and $\gamma_{j+1} := \tilde{g}(\alpha, r^*)$.

Proof of Theorem 3.4.10. We argue completeness, then soundness, and then triviality of the predicate f_n.

Completeness. Fix $\mathbb{x} = (\mathbb{F}, H, n, d, p_0, \ldots, p_n, \gamma) \in L_{\mathsf{Shmr}}$ and $\mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}$ such that $f_j(\mathbb{x}, \mathbb{z}_1) = \cdots = f_j(\mathbb{x}, \mathbb{z}_\mathsf{t}) = 1$ where $\mathbb{z}_i := (r_1^{(i)}, \ldots, r_j^{(i)}, \gamma_j^{(i)})$. We argue that

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}) \rangle \,\right] = 1\,.$$

For every $i \in [\mathsf{t}]$, since $f_j(\mathbb{x}, \mathbb{z}_i) = 1$, we have that

$$\begin{aligned} p_j(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathsf{t}}(i)) \cdot \prod_{\alpha_{j+1} \in H} g(i, \alpha_{j+1}) &= p_j(r_1^{(i)}, \ldots, r_j^{(i)}) \cdot \prod_{\alpha_{j+1} \in H} h_{j+1}(r_1^{(i)}, \ldots, r_j^{(i)}, \alpha_{j+1}) \\ &= h_j(r_1^{(i)}, \ldots, r_j^{(i)}) \\ &= \gamma_j^{(i)}\,. \end{aligned}$$

Therefore the reduction verifier $\mathbf{V}_{\mathsf{IR}}$ does not reject in the consistency test (in Item 1). Moreover, by the definition of g, for every α and r^*,

$$\begin{aligned} h_{j+1}(r'_1, \ldots, r'_j, r^*) &= h_{j+1}(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathsf{t}}(\alpha), r^*) \\ &= g(\alpha, r^*) \\ &= \gamma_{j+1}\,. \end{aligned}$$

Therefore we have $f_i(\mathbb{x}, \mathbb{z}') = 1$ with probability 1.

Soundness. Fix $\mathbb{x} = (\mathbb{F}, H, n, d, p_0, \ldots, p_n, \gamma) \notin L_{\mathsf{Shmr}}$ and $\mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}$ where $\mathbb{z}_i := (r_1^{(i)}, \ldots, r_j^{(i)}, \gamma_j^{(i)})$. Suppose that $f_j(\mathbb{x}, \mathbb{z}_i) = 0$ for some $i \in [\mathsf{t}]$. Fix a malicious reduction prover $\tilde{\mathbf{P}}_{\mathsf{IR}}$. We show that

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}) \rangle \,\right] \leq \frac{d \cdot (1 + n \cdot (\mathsf{t} - 1))}{|\mathbb{F}|}\,.$$

If $\tilde{g}$ does not pass the consistency test (in Item 1) then the reduction verifier rejects. Thus, we can assume that $\tilde{g}$ passes the consistency test. That is, for every $i \in [\mathsf{t}]$ it holds that

$$p_j(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathsf{t}}(i)) \cdot \prod_{\alpha_{j+1} \in H} \tilde{g}(i, \alpha_{j+1}) = \gamma_j^{(i)}\,.$$

Let g be defined as in Equation 3.4 with respect to $\mathbb{x}$ and $\mathbb{z}_1, \ldots, \mathbb{z}_\mathsf{t}$.

We have that $g \not\equiv \tilde{g}$ due to the fact that $f_j(\mathbb{x}, \mathbb{z}_i) = 0$, which implies that

$$\begin{aligned} p_j(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathsf{t}}(i)) \cdot \prod_{\alpha_{j+1} \in H} g(i, \alpha_{j+1}) &= h_j(r_1^{(i)}, \ldots, r_j^{(i)}) \\ &\neq \gamma_j^{(i)} \\ &= p_j(I_{\mathbb{z}_1,\ldots,\mathbb{z}_\mathsf{t}}(i)) \cdot \prod_{\alpha_{j+1} \in H} \tilde{g}(i, \alpha_{j+1})\,. \end{aligned}$$

Thus, by applying the Schwartz–Zippel lemma, and recalling that g and $\tilde{g}$ both have degree $j \cdot d \cdot (\mathrm{t}-1)$ in their first variable and d in their second variable, we get:

$$\Pr_{\alpha,r^*}[\, g(\alpha,r^*) = \tilde{g}(\alpha,r^*)\,] \leq \frac{d\cdot(1+j\cdot(\mathrm{t}-1))}{|\mathbb{F}|} \leq \frac{d\cdot(1+n\cdot(\mathrm{t}-1))}{|\mathbb{F}|}\,.$$

For any choice of α and r^* such that $g(\alpha,r^*) \neq \tilde{g}(\alpha,r^*)$, we have that

$$h_{j+1}(r'_1,\ldots,r'_j,r^*) = g(\alpha,r^*) \neq \tilde{g}(\alpha,r^*) = \gamma_{j+1}\,.$$

and so $f_{j+1}(\mathbb{x},(r'_1,\ldots,r'_j,r^*,\gamma_{j+1})) = 0$ with probability at least $1 - \frac{d\cdot(1+n\cdot(\mathrm{t}-1))}{|\mathbb{F}|}$.

Triviality. By definition, $f_n(\mathbb{x},\mathbb{z}) = f_n((\mathbb{F},H,n,d,p_0,\ldots,p_n,\gamma),(r_1,\ldots,r_n,\gamma_n)) = 1$ if and only if

$$h_j(r_1,\ldots,r_n) \triangleq p_n(r_1,\ldots,r_n) = \gamma\,,$$

which can be efficiently verified by querying p_n once.

Complexity measures. We discuss complexity measures of the protocol $(\mathbf{P}_{\mathsf{IR}},\mathbf{V}_{\mathsf{IR}})$.

- *Output length.* The output is a list of $j+1 \leq n+1$ field elements. Thus, the output length is at most $O(n\cdot\log|\mathbb{F}|)$, which is a fixed polynomial in the input length.
- *Rounds.* The interactive reduction has one round.
- *Communication complexity.* The prover sends g which is bivariate and has degree at most $n\cdot d\cdot(\mathrm{t}-1)$ in its first variable and d in its second variable. This requires sending $O(n\mathrm{t}d^2)$ field elements. The verifier sends two field elements. Thus the communication complexity of the protocol is $O(n\mathrm{t}d^2\log|\mathbb{F}|)$ bits.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{IR}}$ uses $2\log|\mathbb{F}|$ random bits.
- *Verifier running time.* $\mathbf{V}_{\mathsf{IR}}$ runs in time $\mathrm{poly}(n,\log|\mathbb{F}|) = \mathrm{poly}(|\mathbb{x}|)$.
- *Final predicate time.* Computing the final predicate requires one call to p_n.

□

3.5 IOPs from interactive reducibility

We show how to use interactive reducibility to construct IOPs with small query complexity. In Section 3.5.1, we show that every language with a bounded-output-length interactive reduction (with good enough soundness) has a round-query IOP with round-query complexity $O(1)$. Recall that a round-query IOP with round-query complexity $\mathsf{q}_{\mathsf{rnd}}$ is an IOP in which the verifier reads $\mathsf{q}_{\mathsf{rnd}}$ rounds (prover and verifier messages) in their entirety. In Section 3.5.2, we show that, while the interactive reduction for any IP (in Section 3.4.3.1) does not have bounded output length, a similar claim can be made for any k-round IP; in this case, the resulting verifier queries $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$ rounds.

Combining Theorem 3.5.3 and Theorem 3.9.1 (for transforming round-query IOPs into binary IOPs), we obtain the following corollary showing that languages with interactive reducibility have binary IOPs with constant query complexity.

Corollary 3.5.1 (restatement of Theorem 2)**.** *Let $\mathcal{R}$ be a relation with a bounded-output-length interactive reduction with well-behaved soundness error, k_{IR} predicates, and ℓ_{IR} rounds. Then $\mathcal{R}$ has a non-adaptive public-coin IOP with the parameters below.*

IOP for $\mathcal{R}$	
Rounds	$O(\ell_{\mathsf{IR}} \cdot \mathsf{k}_{\mathsf{IR}})$
Proof length	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Alphabet size	2
Queries	$O(\ell_{\mathsf{IR}})$
Interaction randomness	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Decision randomness	$O(\log\lvert\mathbb{x}\rvert)$
Soundness error	$O(1)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

Proof. We first amplify the soundness error of the interactive reduction using Theorem 3.4.5 until we have $\binom{2\cdot\mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} \cdot \mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, 2\cdot\mathsf{k}_{\mathsf{IR}}) < \frac{1}{2}$. Since $\varepsilon_{\mathsf{IR}}$ is well-behaved and $\binom{2\cdot\mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} < 2^{2\mathsf{k}_{\mathsf{IR}}}$, this incurs only a polynomial overhead. We then apply Theorem 3.5.3 with parameter $\tau = 2$ to get an IOP with $O(\ell_{\mathsf{IR}} \cdot \mathsf{k}_{\mathsf{IR}})$ rounds in which the verifier reads only $O(\ell_{\mathsf{IR}})$ of the rounds with alphabet size $2^{\mathrm{poly}(|\mathbb{x}|)}$. We then apply the transformation described Theorem 3.9.1 to transform this IOP into a binary one in which the verifier queries $O(\ell_{\mathsf{IR}})$ rounds and makes $O(1)$ queries to each round (including its own random messages). □

Similarly, we get the following corollary for any public-coin IP.

Corollary 3.5.2 (restatement of Theorem 1)**.** *Let $\mathcal{R}$ be a relation with a non-adaptive k_{IP}-round public-coin IP. Then $\mathcal{R}$ has an IOP with the parameters below.*

IOP for $\mathcal{R}$	
Rounds	k_{IP}
Proof length	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Alphabet size	2
Queries	$\max\{O(1), O(\mathsf{k}_{\mathsf{IP}}/\log\lvert\mathbb{x}\rvert)\}$
Interaction randomness	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Decision randomness	$O(\log\lvert\mathbb{x}\rvert)$
Soundness error	$O(1)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

Proof. Given an IP for the relation $\mathcal{R}$, we first apply the Babai–Moran transformation [BM88] to reduce the number of rounds by a constant fraction (enough to counteract the constant added to the number of rounds by the next two transformations). Next, we apply Theorem 3.5.7 when setting $m := \max\{O(1), O(\mathsf{k}'_{\mathsf{IP}}/\log\lvert\mathbb{x}\rvert)\}$ (where $\mathsf{k}'_{\mathsf{IP}}$ is the number of rounds of the IP following the Babai–Moran transformation) and then apply Theorem 3.9.1 to the resulting IOP. □

3.5.1 Round-query IOPs from bounded-output-length interactive reductions

We show that any relation with a bounded-output-length interactive reduction has a round-query IOP with polynomial proof length with small round-query complexity.

Theorem 3.5.3. *Let $\tau \in \mathbb{N}$ and let $\mathcal{R}$ be a relation with a bounded-output-length interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ with well-behaved soundness error. Then $\mathcal{R}$ has a public-coin round-query IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with the parameters indicated below.*

Interactive reduction for $\mathcal{R}$	
Number of predicates	k_{IR}
Rounds	ℓ_{IR}
Communication	l_{IR}
Output length	s_{IR}
Randomness	r_{IR}
Soundness error	$\varepsilon_{\mathsf{IR}}$
Verifier running time	$\mathsf{vt}_{\mathsf{IR}}$
Final predicate time	$\mathsf{ft}_{\mathsf{IR}}$

$\longrightarrow$

Round-query IOP for $\mathcal{R}$	
Rounds	$\tau \cdot \mathsf{k}_{\mathsf{IR}} \cdot \ell_{\mathsf{IR}}$
Proof length (per round)	$\tau \cdot \mathsf{k}_{\mathsf{IR}}^2 \cdot (\mathsf{s}_{\mathsf{IR}} + \mathsf{l}_{\mathsf{IR}})$
Round queries	$O(\ell_{\mathsf{IR}})$
Interaction randomness	$O(\tau \cdot \mathsf{k}_{\mathsf{IR}} \cdot \mathsf{r}_{\mathsf{IR}})$
Decision randomness	$O(\log(\tau \cdot \mathsf{k}_{\mathsf{IR}}))$
Soundness error	$\max\{1/\tau, \binom{\tau\cdot\mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} \cdot \mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})\}$
Verifier running time	$\mathrm{poly}(\tau, \mathsf{k}_{\mathsf{IR}}, \mathsf{s}_{\mathsf{IR}}, \mathsf{l}_{\mathsf{IR}}, \mathsf{vt}_{\mathsf{IR}}, \mathsf{ft}_{\mathsf{IR}})$

Construction 3.5.4. Let $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ be an interactive reduction for $\mathcal{R}$ with k_{IR} predicates.

The IOP prover $\mathbf{P}_{\mathsf{IOP}}$ receives as input an instance $\mathbb{x}$ and witness $\mathbb{w}$, and the IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. For every $i \in \{0, \ldots, \tau \cdot \mathsf{k}_{\mathsf{IR}}\}$, $\mathbf{P}_{\mathsf{IOP}}$ defines the $(\mathsf{k}_{\mathsf{IR}} + 1)$-entry array A_i as follows

$$A_i[j] := \begin{cases} \{\perp\} & \text{if } j = 0 \\ \emptyset & \text{if } j \in \{1, \ldots, \mathsf{k}_{\mathsf{IR}}\} \end{cases}.$$

   The set $A_i[j]$ will be used to store all of the instances corresponding to f_j collected by iteration i of the protocol.

2. For $i = 1, \ldots, \tau \cdot \mathsf{k}_{\mathsf{IR}}$:
   (a) $\mathbf{P}_{\mathsf{IOP}}$ sends A_{i-1} to $\mathbf{V}_{\mathsf{IOP}}$.
   (b) Do in parallel for every $j \in [\mathsf{k}_{\mathsf{IR}}]$ such that $A_{i-1}[j-1] \neq \emptyset$:
      i. Execute the *interaction phase* of:

$$\mathbb{z}'_{i,j} \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, A_{i-1}[j-1]), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1]) \rangle .$$

         Here the verifier only sends messages and does not verify the correctness of the execution. Hence $\mathbb{z}'_{i,j}$ is computed only by the prover. Moreover, the verifier's random messages are shared among the parallel executions of $\langle \mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}} \rangle$ over each choice of $j \in [\mathsf{k}_{\mathsf{IR}}]$.
      ii. $\mathbf{P}_{\mathsf{IOP}}$ sets $A_i[j] := A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$.

3. $\mathbf{P}_{\mathsf{IOP}}$ sends $A_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}$. Additionally, for every $i \in \{0, \ldots, \tau \cdot \mathsf{k}_{\mathsf{IR}}\}$, it sends $B_i := A_i$. This concludes the interaction phase.

4. In the decision phase, $\mathbf{V}_{\mathsf{IOP}}$ is given oracle access to a transcript with the following structure:

$$\left(A_0, \left\{\mathsf{tr}_{1,j}\right\}_{j \in [\mathsf{k}_{\mathsf{IR}}]}, A_1, \cdots, \left\{\mathsf{tr}_{\tau \cdot \mathsf{k}_{\mathsf{IR}}, j}\right\}_{j \in [\mathsf{k}_{\mathsf{IR}}]}, A_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}, (B_0, \cdots, B_{\tau \cdot \mathsf{k}_{\mathsf{IR}}})\right), \tag{3.5}$$

   where $\mathsf{tr}_{i,j}$ corresponds to the interaction between the prover and verifier in the j-th parallel execution of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ on the i-th iteration. $\mathbf{V}_{\mathsf{IOP}}$ performs the checks below.
   (a) *Subset consistency.* Read the arrays $B_0, B_1, \ldots, B_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}$ in their entirety. For every $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$ and $j \in \{0, \ldots, \mathsf{k}_{\mathsf{IR}}\}$ check that $B_{i-1}[j] \subseteq B_i[j]$.
   (b) *Transcript consistency.* Sample a random $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$. Read the arrays A_{i-1} and A_i sent by $\mathbf{P}_{\mathsf{IOP}}$ and the entire ℓ_{IR}-round interaction $\left\{\mathsf{tr}_{i,j}\right\}_{j \in [\mathsf{k}_{\mathsf{IR}}]}$ between the prover and the verifier done during the i-th iteration of the the interaction phase.
      i. Check that $A_{i-1} = B_{i-1}$ and $A_i = B_i$.

ii. For every $j \in [\mathsf{k}_{\mathsf{IR}}]$, check that $A_i[j] = A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$ where

$$\mathbb{z}'_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1]; \mathsf{tr}_{i,j})\,.$$

(If $\mathbf{V}_{\mathsf{IR}}$ rejects then $\mathbf{V}_{\mathsf{IOP}}$ immediately rejects.)

(c) *Final predicate holds.* Check that $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}) = 1$ for every $\mathbb{z} \in B_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}[\mathsf{k}_{\mathsf{IR}}]$.

Proof of Theorem 3.5.3. We prove completeness and soundness; then we analyze the complexity measures of the protocol.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. We consider each of $\mathbf{V}_{\mathsf{IOP}}$'s checks, and show that $\mathbf{P}_{\mathsf{IOP}}$'s messages satisfy each check with probability 1.

(a) *Subset consistency.* $\mathbf{P}_{\mathsf{IOP}}$ sets, in each round i and for every j, $A_i[j] := A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$ where $\mathbb{z}'_{i,j}$ is the output of the interactive reduction. Since $\mathbf{P}_{\mathsf{IOP}}$ is honest, $B_{i-1} = A_{i-1}$ and $B_i = A_i$. Together this implies that $B_{i-1}[j] \subseteq B_i[j]$ and so $\mathbf{V}_{\mathsf{IOP}}$'s subset consistency check passes.

(b) *Transcript consistency.* The conditions established in the previous item also directly imply that $\mathbf{V}_{\mathsf{IOP}}$'s transcript consistency check passes.

(c) *Final predicate holds.* Since $B_i := A_i$, if A_i contains only round instances for which the predicate holds, then so does B_i, and $\mathbf{V}_{\mathsf{IOP}}$ accepts. We prove by induction on $i \in \{0, \ldots, \tau \cdot \mathsf{k}_{\mathsf{IR}}\}$ that for every $j \in \{0, \ldots, \mathsf{k}_{\mathsf{IR}}\}$ and $\mathbb{z} \in A_i[j]$ it holds that $f_j(\mathbb{x}, \mathbb{z}) = 1$.

First we consider the case when $j = 0$ (and any i). For every i, we have that $A_i[0] = \{\perp\}$. Since $\mathbb{x} \in L(\mathcal{R})$, we have that $f_j(\mathbb{x}, \perp) = 1$.

Next we consider the case when $j > 0$.

- $i = 0$: For every $j \in [\mathsf{k}_{\mathsf{IR}}]$ we set $A_0[j] = \emptyset$ and so the property holds (trivially) for these values of j.
- $i > 0$: Suppose that, with probability 1, for every $j \in [\mathsf{k}_{\mathsf{IR}}]$ and $\mathbb{z}_i \in A_i[j]$ it holds that $f_j(\mathbb{x}, \mathbb{z}_i) = 1$. We argue that, with probability 1, for every $j \in [\mathsf{k}_{\mathsf{IR}}]$ and $\mathbb{z}_{i+1} \in A_{i+1}[j]$, it holds that $f_j(\mathbb{x}, \mathbb{z}_{i+1}) = 1$. Fix $j > 0$. Since for every $\mathbb{z}_i \in A_i[j]$, $f_j(\mathbb{x}, \mathbb{z}_i) = 1$ and $A_i[j] := A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$, we need only show that $f(\mathbb{x}, \mathbb{z}'_{i,j}) = 1$. $\mathbb{z}'_{i,j}$ is generated by applying the interactive reduction with inputs $\mathbb{x}$ and $\{\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}_{i,j}}\} := A_{i-1}[j-1]$. Since for every $\mathbb{z}' \in A_{i-1}[j-1]$, $f_{j-1}(\mathbb{x}, \mathbb{z}') = 1$, by completeness of the interactive reduction we have that

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}'_{i,j}) = 1 \;\middle|\; \mathbb{z}'_{i,j} \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}_{i,j}}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}_{i,j}}) \rangle \,\right] = 1\,.$$

Soundness. Fix $\mathbb{x} \notin \mathcal{R}(L)$ and an IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. We show that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\max\{1/\tau, \binom{\tau \cdot \mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} \cdot \mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})\}$. Consider a transcript tr of the interaction of the IOP with the structure described in Equation 3.5. For fixed tr we say that an index $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$ is *consistent* if:

- $A_{i-1} = B_{i-1}$ and $A_i = B_i$ and,
- For every $j \in [\mathsf{k}_{\mathsf{IR}}]$: $A_i[j] = A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$ where $\mathbb{z}'_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j]; \mathsf{tr}_{i,j})$ (if $\mathbf{V}_{\mathsf{IR}}$ rejects then i is not consistent).

We give two claims analyzing the probability that the verifier accepts relative to how many indices of the protocol are consistent. Theorem 3.5.5 shows that in any execution of the protocol with less than k_{IR} consistent indices, $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $1/\tau$. By Theorem 3.5.6, for any fixed set of k_{IR} indices, conditioned on the interaction outputting a transcript in which these indices are consistent, $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$. Since there are $\binom{\tau \cdot \mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}}$ choices of k_{IR} indices, we can conclude that, conditioned on the transcript having at least k_{IR} consistent indices, $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\binom{\tau \cdot \mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} \cdot \mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$. Putting the two cases together, we conclude that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\max\{1/\tau, \binom{\tau \cdot \mathsf{k}_{\mathsf{IR}}}{\mathsf{k}_{\mathsf{IR}}} \cdot \mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})\}$.

Claim 3.5.5. *Conditioned on the transcript* tr *generated by* $\langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}}(\mathbb{x}) \rangle$ *being consistent with less than* k_{IR} *indices,* $\mathbf{V}_{\mathsf{IOP}}$ *accepts with probability at most* $1/\tau$ *(over its decision randomness).*

Proof. $\mathbf{V}_{\mathsf{IOP}}$ accepts only if its choice of $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$ in Item 4b is consistent. Since there are at most k_{IR} consistent indices, the probability of the verifier sampling one of these indices is at most $1/\tau$. □

Claim 3.5.6. *Fix indices* $1 \leq i_1 < \cdots < i_{\mathsf{k}_{\mathsf{IR}}} \leq \tau \cdot \mathsf{k}_{\mathsf{IR}}$*. Then, conditioned on the transcript* tr *generated by* $\langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}}(\mathbb{x}) \rangle$ *being consistent with respect to indices* $i_1, \ldots, i_{\mathsf{k}_{\mathsf{IR}}}$*,* $\mathbf{V}_{\mathsf{IOP}}$ *accepts with probability at most* $\mathsf{k}_{\mathsf{IR}} \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$ *(over its interaction randomness).*

Proof. Let $i_0 := 0$. We show by induction that for every $j \in \{0, \ldots, \mathsf{k}_{\mathsf{IR}}\}$ there exists $\mathbb{z} \in A_{i_j}[j]$ such that $f_j(\mathbb{x}, \mathbb{z}) = 0$ except with probability $j \cdot \varepsilon_{\mathsf{IR}}$. This implies the statement, since if $f_{\mathsf{k}_{\mathsf{IR}}}(\mathbb{x}, \mathbb{z}) = 0$, then $\mathbf{V}_{\mathsf{IOP}}$ rejects in Item 4c.

This is immediate for $j = 0$, since $\mathbb{x} \notin L(\mathcal{R})$ and so $f_0(\mathbb{x}, \mathbb{z}) = 0$ for any $\mathbb{z}$. Fix some $j > 0$ and suppose that there exists $\mathbb{z} \in A_{i_{(j-1)}}[j-1]$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}) = 0$ (which happens with probability $1 - (j-1) \cdot \varepsilon_{\mathsf{IR}}$). We first show that $\mathbb{z} \in A_{i_j}[j-1]$. This can be seen by the following:

$$\mathbb{z} \in A_{i_{j-1}}[j-1] = B_{i_{j-1}}[j-1] \subseteq B_{i_j-1}[j-1] = A_{i_j-1}[j-1]\,,$$

where the equalities are due to the fact that the indices $i_{(j-1)}$ and i_j are consistent. The containment of $B_{i_{(j-1)}}[j-1]$ in $B_{i_j-1}[j-1]$ is due to the fact that $i_{(j-1)} \leq i_j - 1$ and that, in order for the verifier to accept the transcript, the check in Item 4a must pass.

This implies that for any $\tilde{\mathbf{P}}_{\mathsf{IR}}$ (and in particular for whatever $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ does in this stage):

$$\Pr\left[f_j(\mathbb{x}, \mathbb{z}'_{i_j,j}) = 1 \,\middle|\, \mathbb{z}'_{i_j,j} \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i_j-1}[j-1]) \rangle \right] \leq \varepsilon_{\mathsf{IR}}(\mathbb{x}, |A_{i_j-1}[j-1]|) \leq \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})\,,$$

where the final inequality is true since $|A_{i_j-1}[j-1]| \leq \tau \cdot \mathsf{k}_{\mathsf{IR}}$, when additionally noting that $\varepsilon_{\mathsf{IR}}$ is well-behaved. Using again the fact that index i_j is consistent, we have

that $A_{i_j}[j] := A_{i_j-1}[j] \cup \{\mathbb{z}'_{i_j,j}\}$ where $\mathbb{z}'_{i_j,j}$ is a round instance generated by the actual interaction between $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ and $\mathbf{V}_{\mathsf{IOP}}$ during the j-th parallel execution of the interactive reduction in iteration i_j. Therefore, we have that, conditioned there existing $\mathbb{z} \in A_{i_{(j-1)}}[j-1]$ where $f_{j-1}(\mathbb{x}, \mathbb{z}) = 0$, there exists $\mathbb{z}'_{i_j,j} \in A_{i_j}[j]$ where $f_j(\mathbb{x}, \mathbb{z}'_{i_j,j}) = 0$ except with probability $\varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$. By the induction hypothesis, the condition happens with probability at least $1 - (j-1) \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$ and so we have that with probability at least $1 - j \cdot \varepsilon_{\mathsf{IR}}(\mathbb{x}, \tau \cdot \mathsf{k}_{\mathsf{IR}})$, there exists $\mathbb{z}'_{i_j,j} \in A_{i_j}[j]$ where $f_j(\mathbb{x}, \mathbb{z}'_{i_j,j}) = 0$. □

Complexity measures. We analyze the efficiency parameters of the IOP:

- *Proof length.* The largest message sent by $\mathbf{P}_{\mathsf{IOP}}$ is the final round, which contains all of the instances generated in the previous interaction. In each of the $\tau \cdot \mathsf{k}_{\mathsf{IR}}$ rounds, one instance is added to the list sent to the verifier for every $j \in [\mathsf{k}_{\mathsf{IR}}]$. The output of the interactive reduction has length at most s_{IR}. Thus this requires sending $O(\tau \cdot \mathsf{k}_{\mathsf{IR}}^2 \cdot \mathsf{s}_{\mathsf{IR}})$ bits. In every execution of the interactive reduction an additional l_{IR} bits are sent, for a total of $\mathsf{k}_{\mathsf{IR}} \cdot \mathsf{l}_{\mathsf{IR}}$ per round. Thus the final proof is $O(\tau \cdot \mathsf{k}_{\mathsf{IR}}^2 \cdot (\mathsf{s}_{\mathsf{IR}} + \mathsf{l}_{\mathsf{IR}}))$ bits long.
- *Round complexity.* The IOP contains $\tau \cdot \mathsf{k}_{\mathsf{IR}}$ executions of the ℓ_{IR}-round interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$. Therefore the IOP has $\tau \cdot \mathsf{k}_{\mathsf{IR}} \cdot \ell_{\mathsf{IR}}$ rounds.
- *Round queries.* $\mathbf{V}_{\mathsf{IOP}}$ reads in its entirety the last message in the protocol, reads A_{i-1} and A_i, and reads the execution of the interactive reduction between blocks $i-1$ and i. Therefore it reads $O(\ell_{\mathsf{IR}})$ rounds.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{IOP}}$ generates r_{IR} bits for every separate $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$ since we share verifier randomness between each of the parallel executions of $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$. The interaction randomness is therefore $O(\tau \cdot \mathsf{k}_{\mathsf{IR}} \cdot \mathsf{r}_{\mathsf{IR}})$ bits. There are $O(\log(\tau \cdot \mathsf{k}_{\mathsf{IR}}))$ bits of decision randomness because $\mathbf{V}_{\mathsf{IOP}}$ samples a random $i \in [\tau \cdot \mathsf{k}_{\mathsf{IR}}]$.
- *Verifier running time.* $\mathbf{V}_{\mathsf{IOP}}$ checks all of the arrays $B_1, \ldots, B_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}$ for consistency, executes the interactive reduction k_{IR} times (each in time $\mathsf{vt}_{\mathsf{IR}}$), and verifies the final predicate of at most $\tau \cdot \mathsf{k}_{\mathsf{IR}}$ elements in $B_{\tau \cdot \mathsf{k}_{\mathsf{IR}}}$ (each in time $\mathsf{ft}_{\mathsf{IR}}$). Overall $\mathbf{V}_{\mathsf{IOP}}$ runs in time $\mathrm{poly}(\tau, \mathsf{k}_{\mathsf{IR}}, \mathsf{s}_{\mathsf{IR}}, \mathsf{l}_{\mathsf{IR}}, \mathsf{vt}_{\mathsf{IR}}, \mathsf{ft}_{\mathsf{IR}})$.
- *Adaptivity.* $\mathbf{V}_{\mathsf{IOP}}$ is non-adaptive.

□

3.5.2 Round-query IOPs from public-coin IPs

The interactive reduction for any public-coin IP described in Section 3.4.3.1 does not have bounded output length. Hence we cannot use it directly in Theorem 3.5.3. Nonetheless, the output length grows slowly enough to achieve weaker results. While the transformation is essentially identical to the protocol described in Theorem 3.5.4, we describe it explicitly.

Theorem 3.5.7. *Let $\mathcal{R}$ be a relation with a public-coin IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ with k_{IP} rounds. Let $\tau, m \geq 1$ be parameters where m divides k_{IP}. Then $\mathcal{R}$ has a round-query IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with the parameters indicated below.*

IP for $\mathcal{R}$	
Rounds	k_{IP}
Total communication	l_{IP}
Randomness per round	r_{IP}
Round-by-round soundness	β_{rbr}
Verifier running time	$\mathsf{vt}_{\mathsf{IP}}$

$\longrightarrow$

Round-query IOP for $\mathcal{R}$	
Rounds	$\tau \cdot \mathsf{k}_{\mathsf{IP}}$
Proof length (per round)	$O(\mathsf{l}_{\mathsf{IP}} \cdot 2^{\tau \cdot \mathsf{k}_{\mathsf{IP}}/m})$
Round queries	$O(m)$
Interaction randomness	$O(\tau \cdot \mathsf{r}_{\mathsf{IP}} \cdot \mathsf{k}_{\mathsf{IP}})$
Decision randomness	$O(\log(\tau \cdot \mathsf{k}_{\mathsf{IP}}/m))$
Soundness error	$\max\{1/\tau, \binom{\tau \cdot \mathsf{k}_{\mathsf{IP}}/m}{\mathsf{k}_{\mathsf{IP}}/m} \cdot m \cdot \mathsf{k}_{\mathsf{IP}} \cdot \beta_{\mathrm{rbr}}\}$
Verifier running time	$\mathrm{poly}(2^{\tau \cdot \mathsf{k}_{\mathsf{IP}}/m}, \mathsf{l}_{\mathsf{IP}}, \mathsf{vt}_{\mathsf{IR}})$

Construction 3.5.8. Let $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ be a public-coin IP with k_{IP} rounds and where the verifier message at each round is r_{IP} bits long. Let $\tau > 1$ be a parameter. On input $\mathbb{x}$ and with witness $\mathbb{w}$ the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ is as follows:

- Prover sets $S_0 := \{\emptyset\}$ (i.e., S_0 contains only the empty transcript).
- For $i = 1$ to $\tau \cdot \mathsf{k}_{\mathsf{IP}}/m$:
  - Prover sends S_{i-1}.
  - For $j \in [m]$:
    1. Verifier chooses and sends $\alpha_{i,j} \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{IP}}}$ uniformly at random.
    2. For every $\mathsf{tr} \in S_{i-1}$, prover sends $a_{\mathsf{tr},j} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathbb{w}, \mathsf{tr}||\alpha_{i,1}||a_{\mathsf{tr},1}|| \ldots ||\alpha_{i,j})$.
  - Prover sets $S_i := S_{i-1} \cup \{(\mathsf{tr}||\alpha_{i,1}||a_{\mathsf{tr},1}|| \ldots ||\alpha_{i,m}||a_{i,m})\}_{\mathsf{tr} \in S_{i-1}}$.
- For every $i \in [\tau \cdot \mathsf{k}_{\mathsf{IP}}/m]$, the Prover sends $T_i := S_i$.
- Verifier accepts if and only if:
  1. *Subset consistency:* For every $i \in [\tau \cdot \mathsf{k}_{\mathsf{IP}}/m]$, $T_{i-1} \subseteq T_i$.
  2. *Transcript consistency:* Choose a random $i \in [\tau \cdot \mathsf{k}_{\mathsf{IP}}/m]$. Check that $S_{i-1} = T_{i-1}$ and $S_i = T_i$. Additionally, assert that for every $\mathsf{tr} \in S_{i-1}$, there are some messages $a_{\mathsf{tr},1} \ldots, a_{\mathsf{tr},m}$ such that $(\mathsf{tr}||\alpha_{i,1}||a_{\mathsf{tr},1}|| \ldots ||\alpha_{i,m}||a_{i,m}) \in S_i$, where $\alpha_{i,1} \ldots, \alpha_{i,m}$ are the verifier messages sent during the i-th round of interaction.
  3. *Membership:* Every transcript $\mathsf{tr} \in T_{\tau \cdot \mathsf{k}_{\mathsf{IP}}/m}$ that is complete (i.e., contains messages for all k_{IP} rounds of the IP) has $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}) = 1$.

Proof of Theorem 3.5.7. The proofs of completeness and soundness are identical to those for Theorem 3.5.3, using the interactive reduction for IPs described in Theorem 3.4.7. We therefore only need to analyze the complexity measures of the resulting IOP.

- *Proof length.* The proof length of the protocol is $O(\mathsf{l}_{\mathsf{IP}} \cdot 2^{\tau \cdot \mathsf{k}_{\mathsf{IP}}/m})$ bits.
- *Rounds.* The protocol has $\tau \cdot \mathsf{k}_{\mathsf{IP}}$ rounds.
- *Round queries.* The verifier reads $O(m)$ rounds.
- *Randomness complexity.* The IOP verifier uses $O(\tau \cdot \mathsf{r}_{\mathsf{IP}} \cdot \mathsf{k}_{\mathsf{IP}})$ bits of interaction randomness and $O(\log(\tau \cdot \mathsf{r}_{\mathsf{IP}}/m))$ bits of decision randomness .
- *Verifier running time.* The verifier running time is $\mathrm{poly}(2^{\tau \cdot \mathsf{k}_{\mathsf{IP}}}, \mathsf{l}_{\mathsf{IP}}, \mathsf{vt}_{\mathsf{IR}})$.
- *Adaptivity.* $\mathbf{V}_{\mathsf{IOP}}$ is non-adaptive.

□

3.6 Index-decodable PCPs

Let $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be a tuple where $\mathbf{I}_{\mathsf{PCP}}$ (the indexer) is a deterministic algorithm, $\mathbf{P}_{\mathsf{PCP}}$ (the prover) is a deterministic algorithm, $\mathbf{V}_{\mathsf{PCP}}$ (the verifier) is a randomized oracle algorithm, $\mathbf{iD}_{\mathsf{PCP}}$ (the index decoder) and, $\mathbf{wD}_{\mathsf{PCP}}$ (the witness decoder) are (possibly inefficient) deterministic algorithms. PCP is an *index-decodable PCP* for a multi-indexed relation $\mathcal{R}$ with decodability bound κ_{PCP} if the following holds.

- **Completeness.** For every $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$,

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

- **Decodability.** For every $\mathbb{x}$ and strings $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}$, if

$$\Pr_{\alpha}\left[\, \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}}(\mathbb{x};\alpha) = 1 \,\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)$$

  then $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in \mathcal{R}$.

The proofs $\pi_1, \ldots, \pi_{\mathsf{k}}$ are called *indexer proofs* and the proof Π is called the *prover proof*. We refer to Section 3.2.5 for an intuitive overview of this notion and further discussion.

Complexity measures. In addition to the standard complexity measures mentioned in Section 2.1 we consider several additional measures for index-decodable PCPs. All of these complexity measures are implicitly functions of the instance $\mathbb{x}$.

- *Indexer proof length* $\mathsf{l}_{\mathsf{PCP},\mathbf{I}}$: the number of symbols in a single indexer proof π_i.
- *Indexer proof alphabet* $\Sigma_{\mathsf{PCP},\mathbf{I}}$: the alphabet of the indexer proofs.
- *Prover proof length* $\mathsf{l}_{\mathsf{PCP},\mathbf{P}}$: the number of symbols in a single indexer proof Π.
- *Prover proof alphabet* $\Sigma_{\mathsf{PCP},\mathbf{P}}$: the alphabet of the prover's proof.
- *Indexer time* $\mathsf{it}_{\mathsf{PCP}}$: $\mathbf{I}_{\mathsf{PCP}}$ runs in time $\mathsf{it}_{\mathsf{PCP}}$.
- *Index decoding time* $\mathsf{idt}_{\mathsf{PCP}}$: $\mathbf{iD}_{\mathsf{PCP}}$ runs in time $\mathsf{idt}_{\mathsf{PCP}}$.
- *Witness decoding time* $\mathsf{wdt}_{\mathsf{PCP}}$: $\mathbf{wD}_{\mathsf{PCP}}$ runs in time $\mathsf{wdt}_{\mathsf{PCP}}$.

We sometimes refer separately to the number of queries done to the indexer proofs (per proof) and and prover proof. If these are not listed separately, then the number is asymptotically identical.

Remark 3.6.1 (index-decodable IOPs)**.** The definition of index-decodable PCPs can be extended in a straightforward way to allow interaction, thereby obtaining *index-decodable IOPs*. While not used in this thesis, this extended notion is likely to achieve better parameters (e.g., linear proof length) via interactive tools (e.g., interactive proof composition [BCGRS17] instead of non-interactive proof composition as in Section 3.8.1) and is likely to be of further interest. We leave the exploration of this notion to future work.

Remark 3.6.2 (comparison with decodable PCPs)**.** Despite the similar names, index-decodable PCPs and *decodable PCPs* [DH13] are different notions: a decodable PCP is a standard PCP with a "PCP decoder" that list-decodes a random location in the NP witness from a given PCP string.

Prover-robust index-decodable PCPs. Let $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, (\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}), \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be a non-adaptive index-decodable PCP for a multi-indexed relation $\mathcal{R}$. We say that PCP is *prover-robust* with decodability bound κ_{PCP} and robustness σ_{PCP} if for every $\mathbb{x}$ and proofs $\tilde{\mathbf{\Pi}}_{\mathbb{i}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$ and $\tilde{\Pi}$ if

$$\Pr_{\alpha}\left[\exists A' \text{ s.t. } \begin{array}{c}\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_{\mathbb{i}}[Q_{\mathbb{i}}],A')=1\\ \wedge\ \Delta(A',A)\le\sigma_{\mathsf{PCP}}(|\mathbb{x}|)\end{array} \;\middle|\; \begin{array}{l}(Q_{\mathbb{i}},Q_*)\leftarrow\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x},\alpha)\\ A:=\{\,\tilde{\Pi}[q]\mid q\in Q_*\,\}\end{array}\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)\,,$$

where $Q_{\mathbb{i}}$ are the queries made to the indexer proofs and Q_* are the queries made to the prover proof. Then there exists $\mathbb{w}$ such that $(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})) \in \mathcal{R}$.

The notion of robustness essentially says that for any set of proofs whose decoding is not in the relation, with probability κ_{PCP} the Hamming ball of radius σ_{PCP} around the answers to the verifier's queries to the prover proof do not make the verifier accept.

3.7 Basic construction of an index-decodable PCP from PCPPs

We describe a construction of an index-decodable PCP from PCPPs where we achieve query complexity $O(1)$ per oracle, the indexer proofs are over the binary alphabet, and the prover proof are over a large alphabet. Later, in Section 3.8, we additionally achieve a binary alphabet for all proofs.

Theorem 3.7.1. *Let $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ be a multi-indexed relation decidable in* NTIME(T). *Then $\mathcal{R}$ has a non-adaptive index-decodable PCP for $\mathcal{R}$ with the parameters below.*

Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length (per proof)	$O(\lvert\mathbb{i}[i]\rvert)$
Prover proof length	$\mathrm{poly}(T)$
Indexer alphabet size	2
Prover alphabet size	2^{k}
Queries to proofs	$O(1)$
Randomness	$O(\log T)$
Decodability bound	$O(1)$
Indexer running time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Prover running time	$\mathrm{poly}(T)$
Verifier running time	$\mathrm{poly}(\mathsf{k}, \lvert\mathbb{x}\rvert, \log T)$
Index decoding time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Witness decoding time	$\mathrm{poly}(T)$

The construction relies on slight variations of the notion of PCPPs. In Section 3.7.1 we define and construct these notions (based on standard PCPPs), and then in Section 3.7.2 we describe the index-decodable PCP. Theorem 3.7.1 follows by plugging in Theorem 3.7.7 the following two ingredients: (i) the oblivious multi-input PCPP of Theorem 3.7.5; and (ii) the constant-rate and constant-distance error-correcting code in Theorem 2.2.1.

3.7.1 Building blocks

Definition 3.7.2 (Multi-witness PCP of proximity)**.** *A PCP of proximity system for nondeterministic computations is* multi-witness for $(k+1)$-input machines *if the soundness condition is replaced with the following: Let M be a $(k+1)$-input nondeterministic Turing machine, $\mathbb{x}$ be an instance, T be a bound on the running time of M, and $\mathbb{w}_1, \ldots, \mathbb{w}_k$ be candidate witnesses. Suppose that for every set of inputs $\mathbb{w}'_1, \ldots, \mathbb{w}'_k$ such that $M(\mathbb{x}, \mathbb{w}'_1, \ldots, \mathbb{w}'_k) = 1$ there exists i such that $\mathbb{w}_i$ is δ_{PCPP}-far from $\mathbb{w}'_i$. Then for every $\tilde{\Pi}$:*

$$\Pr\left[\, \mathbf{V}_{\mathsf{PCP}}^{\mathbb{w}_1,\ldots,\mathbb{w}_k,\tilde{\Pi}}(M, \mathbb{x}, T) = 1 \,\right] \leq \beta_{\mathsf{PCP}}\,.$$

Definition 3.7.3 (Oblivious PCP of proximity). *Let $\mathcal{M} = \{M_z\}_{z\in\{0,1\}^m}$ be a family of nondeterministic Turing machines and suppose that there exists a machine M' such that for every $\mathbb{x}$ and $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and z: $M'(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k, z) = M_z(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k)$. An oblivious multi-input PCP of proximity system for $\mathcal{M}$ is a PCPP that is complete and sound for proving satisfiability of every M_z, $z \in \{0,1\}^m$. Additionally, the verifier's queries depend only on $\mathcal{M}$ and on the verifier's randomness. In particular, its queries do not depend on z, or on its oracles.*

We now show how to construct these PCPPs from standard PCPPs (with minimal overhead).

Lemma 3.7.4 (Existence of multi-input PCPPs). *There exists a multi-input PCP of proximity for $(k+1)$-input nondeterministic computations where k is a constant with the following parameters:*

Multi-input PCP of Proximity for $M(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k) = 1$ in T time steps	
Proof length	$\text{poly}(T)$
Alphabet size	2
Queries	$O(1)$
Randomness	$O(\log T)$
Proximity	$O(1)$
Soundness error	$O(1)$
Prover running time	$\text{poly}(T)$
Verifier running time	$\text{poly}(\lvert\mathbb{x}\rvert, \log T)$

Proof. Let $(\mathbf{P}_{\mathsf{PCPP}}, \mathbf{V}_{\mathsf{PCPP}})$ be the PCPP system of Theorem 2.1.1 with soundness error β_{PCPP} and proximity parameter δ_{PCPP}, where β_{PCPP} and δ_{PCPP} are small enough constants satisfying $\delta_{\mathsf{PCPP}} \cdot k < 1/3$. If this is not the case, one can first improve proximity using standard techniques (e.g., [RR20, Lemma 8.6]) with minimal overhead. For a witness $\mathbb{w}_i$, let $\mathbb{w}_i^{t_i}$ be the t_i-wise repetition of $\mathbb{w}_i$.

For a set of witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$, let $n_{\mathsf{max}} := \max\{n_i\}$ where n_i is the length of witness i. For every $i \in [k]$ let $t_i := n_{\mathsf{max}}/n_i$. Assume without loss of generality that these values are integers (otherwise, padding each input to the next power of 2 will suffice).

Consider the machine M' that on inputs $\mathbb{x}$ and $(\widehat{\mathbb{w}}_1, \ldots, \widehat{\mathbb{w}}_k) \in \{0,1\}^{k\cdot n_{\mathsf{max}}}$ outputs 1 if and only if the following tests pass:

1. *Encoding validity*: For every i, there exists a string $\mathbb{w}_i$ such that $\widehat{\mathbb{w}} = \mathbb{w}_i^{t_i}$.
2. *Satisfiability*: Let $\mathbb{w}_1, \ldots, \mathbb{w}_k$ be the strings from the previous test. Test that

$$M(\mathbb{x}, \mathbb{w}_1 \ldots, \mathbb{w}_k) = 1\,.$$

The running time of M' is $T' = O(k \cdot \max_i |\mathbb{w}_i|) + T$. We now describe the PCPP. On input a $(k+1)$-input machine M, $\mathbb{x}$ and $(\mathbb{w}_1, \ldots, \mathbb{w}_k) \in \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_k}$ and time-bound T:

- The prover sends $\Pi := \mathbf{P}_{\mathsf{PCPP}}(M', \mathbb{x}, T', \mathbb{w}_1^{t_1}, \ldots, \mathbb{w}_k^{t_k})$.
- The verifier, given oracle access to the input $\mathbb{x}$ and witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and to $\tilde{\Pi}$ emulates the verification of:

$$\mathbf{V}_{\mathsf{PCP}}^{\mathbb{w}_1^{t_1}, \ldots, \mathbb{w}_k^{t_k}, \tilde{\Pi}}(M', \mathbb{x}, T') = 1\,,$$

  as follows: Queries to $\tilde{\Pi}$ are forwarded to the relevant oracle. For a query $q \in [t_i \cdot |\mathbb{w}_i|]$ to $\mathbb{w}_i^{t_i}$, return $\mathbb{w}_i$ at location $(q \mod t_i)$.

Completeness is immediate by the perfect completeness of the PCP of proximity and the fact that the verifier correctly returns queries to $\mathbb{w}_i^{t_i}$. We show soundness by proving the contrapositive. Assume that the verifier accepts with high probability. We show that there must exist strings that satisfy the machine and are individually close to each input. Fix inputs $\mathbb{x}$ and $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and (possibly malicious) proof $\tilde{\Pi}$ such that the verifier, given these inputs and proof, accepts with probability greater than β_{PCPP}. Then we have that with probability greater than β_{PCPP}:

$$\mathbf{V}_{\mathsf{PCP}}^{\mathbb{w}_1^{t_1}, \ldots, \mathbb{w}_k^{t_k}, \tilde{\Pi}}(M', \mathbb{x}, T') = 1\,.$$

This implies, by soundness of the PCPP, that there exist $\widehat{\mathbb{w}}_1', \ldots, \widehat{\mathbb{w}}_k'$ such that

$$M'(\mathbb{x}, \widehat{\mathbb{w}}_1', \ldots, \widehat{\mathbb{w}}_k') = 1\,,$$

and

$$\Delta((\mathbb{w}_1^{t_1}, \ldots, \mathbb{w}_k^{t_k}), (\widehat{\mathbb{w}}_1', \ldots, \widehat{\mathbb{w}}_k')) < \delta_{\mathsf{PCPP}}\,.$$

Since $M'(\mathbb{x}, \widehat{\mathbb{w}}_1', \ldots, \widehat{\mathbb{w}}_k') = 1$, we have that for every i there exists $\mathbb{w}_i'$ such that $\widehat{\mathbb{w}}_i' = \mathbb{w}_i'^{t_i}$. Moreover $M(\mathbb{x}, \mathbb{w}_1', \ldots, \mathbb{w}_k') = 1$.

Notice that for every i: $|\mathbb{w}_i^{t_i}| = |\mathbb{w}_i'^{t_i}| = n_{\mathsf{max}}$. Therefore, by a counting argument,

$$\Delta((\mathbb{w}_1^{t_1}, \ldots, \mathbb{w}_k^{t_k}), (\mathbb{w}_1'^{t_1}, \ldots, \mathbb{w}_k'^{t_k})) < \delta_{\mathsf{PCPP}}\,,$$

implies that for every i: $\Delta(\mathbb{w}_i^{t_i}, \mathbb{w}_i'^{t_i}) < k \cdot \delta_{\mathsf{PCPP}}$. It follows that $\Delta(\mathbb{w}_i, \mathbb{w}_i') < k \cdot \delta_{\mathsf{PCPP}} = O(1)$ since if $\Delta(\mathbb{w}_i, \mathbb{w}_i') = m$, then every one of the differences between the two strings propagates t_i times, and so the relative number of times it occurs is still m: $\Delta(\mathbb{w}_i^{t_i}, \mathbb{w}_i'^{t_i}) = m$.

We now analyze the parameters of the new PCPP. The running time of M' is $O(k \cdot \max_i |\mathbb{w}|) + T = O(T)$. Thus the new prover has running time $\mathrm{poly}(|\mathbb{x}|, T)$, and the proof length is $\mathrm{poly}(T)$ is also the proof length. The verifier uses the same number of random bits as $\mathbf{V}_{\mathsf{PCPP}}$, and runs in time $\mathrm{poly}(|\mathbb{x}|, \log T)$. If the original verifier was non-adaptive, then so is the new verifier. □

Lemma 3.7.5 (Existence of oblivious multi-input PCPPs)**.** *Let $\mathcal{M} = \{M_z\}_{z \in \{0,1\}^m}$ be a family of $(k+1)$-input nondeterministic machines for constant k and suppose there exists nondeterministic machine M' that runs in time T' such that for every $\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k$ and z: $M'(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k, z) = M_z(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k)$. There exists an oblivious multi-input PCP of proximity for $\mathcal{M}$ with the following parameters:*

Oblivious PCP of Proximity for $M_z(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k) = 1$ where M' runs in T' time steps	
Proof length	$\text{poly}(T')$
Alphabet size	2
Queries	$O(1)$
Randomness	$O(T')$
Proximity	$O(1)$
Soundness error	$O(1)$
Prover running time	$\text{poly}(T')$
Verifier running time	$\text{poly}(\lvert\mathbb{x}\rvert, \log T')$

Proof. Let $(\mathbf{P}_{\mathsf{PCPP}}, \mathbf{V}_{\mathsf{PCPP}})$ be the multi-input system of Theorem 3.7.4 and $(\mathbf{Enc}, \mathbf{Dec})$ be the error-correcting code of Theorem 2.2.1.

Consider the $(k+2)$-input machine M'' that on inputs $\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k$ and $\widehat{z}$ outputs 1 if and only if the following tests pass:
1. *Encoding validity*: $\mathbf{Enc}(\mathbf{Dec}(\widehat{z})) = \widehat{z}$.
2. *Satisfiability*: $M'(\mathbb{x}, \mathbb{w}_1, \ldots, \mathbb{w}_k, \mathbf{Dec}(\widehat{z})) = 1$.

Notice that M'' runs in time $T'' = T' + \tilde{O}(|z|)$.

We now describe the PCPP. On explicit inputs M_z, $\mathbb{x}$ and T, and oracle inputs $\mathbb{x}_1, \ldots, \mathbb{x}_k$:
- The prover sends $\Pi := \mathbf{P}_{\mathsf{PCPP}}(M'', \mathbb{x}, T'', (\mathbb{w}_1, \ldots, \mathbb{w}_k, \mathbf{Enc}(z)))$.
- The verifier, given M_z, instance $\mathbb{x}$, T and oracle access to $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and to $\tilde{\Pi}$ computes $\mathbf{Enc}(z)$ and verifies

$$\mathbf{V}_{\mathsf{PCP}}^{(\mathbb{w}_1, \ldots, \mathbb{w}_k, \mathbf{Enc}(z)), \tilde{\Pi}}(M'', \mathbb{x}, T'') = 1\,.$$

Completeness is immediate by the perfect completeness of the multi-input PCPP and the fact that the verifier correctly returns queries to $\widehat{z}$. We show that multi-input soundness holds with respect to the machine M_z. We do this via the contrapositive: we show that if the verifier accepts with high probability, then the witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$ are close to satisfying M_z along with $\mathbb{x}$.

Fix $\mathbb{x}$, z, witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and a proof $\tilde{\Pi}$. Suppose that, given $\mathbb{x}$ and z explicitly and oracle access to $\mathbb{w}_1, \ldots, \mathbb{w}_k$ and $\tilde{\Pi}$, the verifier accepts with probability greater than β_{PCP}. Then we have that there exist $\mathbb{w}'_1, \ldots, \mathbb{w}'_k$ and $\widehat{z}'$ such that: (i) $M''(\mathbb{x}, \mathbb{w}'_1, \ldots, \mathbb{w}'_k, \widehat{z}') = 1$, (ii) For every i: $\Delta(\mathbb{w}_i, \mathbb{w}'_i) < \delta_{\mathsf{PCPP}}$ and, (iii) $\Delta(\widehat{z}, \widehat{z}') < \delta_{\mathsf{PCPP}}$. This must be true since otherwise we have a contradiction to the soundness of the multi-input PCPP. This, in turn, implies that $M'(\mathbb{x}, \mathbb{w}'_1, \ldots, \mathbb{w}'_k, \mathbf{Dec}(\widehat{z}')) = 1$. Since $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{ECC}}$, we have that $\mathbf{Dec}(\widehat{z}') = \mathbf{Dec}(\widehat{z}) = z$.

We now analyze the parameters of the new PCPP. Notice that M'' runs in time $T'' = T' + \tilde{O}(|z|) = O(T')$. Thus the new prover has running time $\text{poly}(|\mathbb{x}|, T')$, and the proof length is $\text{poly}(T')$. The verifier uses the same number of random bits as $\mathbf{V}_{\mathsf{PCPP}}$, and runs in time $\text{poly}(|\mathbb{x}|, \log T')$. Since $\mathbf{V}_{\mathsf{PCPP}}$ is non-adaptive, the queries it makes do not depend on its oracles. This includes the oracle to $\widehat{z}$, and so the new verifier's queries do not depend on z. □

3.7.2 The construction

We begin by giving a construction that achieves all of the parameters of our final index-decodable PCP except that the number of queries made by the verifier to the prover proof is $O(\mathsf{k})$ rather than $O(1)$. We define the machine M_* and family of machines $\mathcal{M} = \{M_i\}_{i\in[\mathsf{k}]}$. Later on, in the construction, we will have a (standard) PCPP proof for satisfiability of M_*, and a proof for an oblivious multi-input PCPP for satisfiability of each of the machines $M_i \in \mathcal{M}$.

Definition 3.7.6. *Let $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ be a multi-indexed relation decidable in nondeterministic time T, and $(\mathbf{Enc}, \mathbf{Dec})$ be an error-correcting code. Let $\mathbb{x}$ be an instance. Define Turing machines M_* and a family of machines $\mathcal{M} = \{M_i\}_{i\in[\mathsf{k}]}$ as follows:*

- *$M_*(\mathbb{x}, \Pi_*) = 1$ if and only if the following tests pass:*
  1. Encoding validity: *Check that* $\mathbf{Enc}(\mathbf{Dec}(\Pi_*)) = \Pi_*$.
  2. Membership: *Check that* $(\mathbb{i}_*[1], \ldots, \mathbb{i}_*[\mathsf{k}], \mathbb{x}, \mathbb{w}_*) \in \mathcal{R}$, *where* $(\mathbb{i}_*[1], \ldots, \mathbb{i}_*[\mathsf{k}], \mathbb{w}_*) := \mathbf{Dec}(\Pi_*)$.

  The machine M_* runs in time $O(T)$.
- *$M_i(\perp, \pi, \Pi_*) = 1$ if and only if the following tests pass:*
  1. Encoding validity: *Check that:*
     - $\mathbf{Enc}(\mathbf{Dec}(\pi)) = \pi$.
     - $\mathbf{Enc}(\mathbf{Dec}(\Pi_*)) = \Pi_*$.
  2. Consistency: *Let* $\tilde{\mathbb{i}}[i] := \mathbf{Dec}(\pi)$ *and* $(\mathbb{i}_*[1], \ldots, \mathbb{i}_*[\mathsf{k}], \mathbb{w}_*) := \mathbf{Dec}(\Pi_*)$. *Check that* $\tilde{\mathbb{i}}[i] = \mathbb{i}_*[i]$.

  From the definition it is easy to see that there exists M' such that $M'(\perp, \pi, \Pi_*, i) = M_i(\perp, \pi, \Pi_*)$ for every π, Π_* where the running time of M' is equal to that of M_i, which is $\tilde{O}(|\mathbb{w}| + |\sum_i |\mathbb{i}[i]|) = O(T)$. This facilitates using this family of machines with an oblivious PCPP.

Theorem 3.7.7. *Let $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ be a multi-indexed relation. Let $\mathsf{PCPP} = (\mathbf{P}_{\mathsf{PCPP}}, \mathbf{V}_{\mathsf{PCPP}})$ be a (oblivious multi-input) PCP of proximity for nondeterministic computations with proximity parameter δ_{PCPP} using a binary alphabet and $(\mathbf{Enc}, \mathbf{Dec})$ be an error-correcting code with distance $\delta_{\mathsf{PCPP}} \leq \delta_{\mathsf{ECC}}$. Then Theorem 3.7.8 is a non-adaptive index-decodable PCP $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ for $\mathcal{R}$ with the parameters below.*

PCPP for satisfiability of machines $\mathcal{M}$ and M_*	
Proof length	$\mathsf{l}_{\mathsf{PCPP}}$
Alphabet size	2
Queries	$\mathsf{q}_{\mathsf{PCPP}}$
Randomness	$\mathsf{r}_{\mathsf{PCPP}}$
Proximity	δ_{PCPP}
Soundness error	β_{PCPP}
Prover running time	$\mathsf{pt}_{\mathsf{PCPP}}$
Verifier running time	$\mathsf{vt}_{\mathsf{PCPP}}$

+

Error correcting code	
Distance	δ_{ECC}
Block length	N_{ECC}
Encoding time	$\mathsf{et}_{\mathsf{ECC}}$
Decoding time	$\mathsf{dt}_{\mathsf{ECC}}$

Index-Decodable PCP for $(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{x},\mathbb{w}) \in \mathcal{R}$	
Indexer proof length (per proof)	$N_{\mathsf{ECC}}(\lvert\mathbb{i}[i]\rvert)$
Prover proof length	$2 \cdot \mathsf{l}_{\mathsf{PCPP}} + N_{\mathsf{ECC}}(\lvert\mathbb{w}\rvert + \sum_{i=1}^{\mathsf{k}} \lvert\mathbb{i}[i]\rvert)$
Indexer alphabet size	2
Prover alphabet size	2^{k}
Queries to indexer proof (per proof)	$\mathsf{q}_{\mathsf{PCPP}}$
$\longrightarrow$ Queries to prover proof	$2 \cdot \mathsf{q}_{\mathsf{PCPP}}$
Randomness	$2 \cdot \mathsf{r}_{\mathsf{PCPP}}$
Decodability bound	β_{PCPP}
Indexer running time	$\mathsf{et}_{\mathsf{ECC}}(\lvert\mathbb{i}[i]\rvert)$
Prover running time	$\mathsf{et}_{\mathsf{ECC}}(\lvert\mathbb{w}\rvert + \sum_{i=1}^{\mathsf{k}} \lvert\mathbb{i}[i]\rvert) + (\mathsf{k}+1) \cdot \mathsf{pt}_{\mathsf{PCPP}}$
Verifier running time	$(\mathsf{k}+1) \cdot \mathsf{vt}_{\mathsf{PCPP}}$
Index decoding time	$\mathsf{dt}_{\mathsf{ECC}}(\lvert\mathbb{i}[i]\rvert)$
Witness decoding time	$\mathsf{dt}_{\mathsf{ECC}}(\lvert\mathbb{w}\rvert + \sum_{i=1}^{\mathsf{k}} \lvert\mathbb{i}[i]\rvert)$

We now describe the construction (see Section 3.2.7.1 for an overview), and then prove the theorems.

Construction 3.7.8. We describe the index-decodable PCP for $\mathcal{R}$. Let T_* and T_i be the running times of T_* and T_i respectively.

- $\mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$: Output $\pi_i := \mathbf{Enc}(\mathbb{i}[i])$.
- $\mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{x},\mathbb{w})$:
  1. Compute $\Pi_* := \mathbf{Enc}(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{w})$.
  2. *Proof of membership*: Generate a proof $\Pi_{\mathsf{mem}} := \mathbf{P}_{\mathsf{PCPP}}(M_*,\mathbb{x},T_*,\Pi_*)$ (using a standard PCPP).
  3. *Proofs of consistency*: For every $i \in [\mathsf{k}]$, compute $\Pi_i := \mathbf{P}_{\mathsf{PCPP}}(M_i,\perp,T_i,(\pi_i,\Pi_*))$ where $\pi_i := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$ (using the oblivious PCPP for $\mathcal{M}$).
  4. *Query bundling*: Let $\Pi_{\mathbb{i}} := \{\Pi_i\}_{i\in[\mathsf{k}]}$ be a proof such that $\Pi_{\mathbb{i}}[q] = (\Pi_1[q],\ldots,\Pi_{\mathsf{k}}[q])$. That is, in location q of $\Pi_{\mathbb{i}}$ write a k-bit symbol consisting of the q-th bit of each of the proofs $\{\Pi_i\}_{i\in[\mathsf{k}]}$.
  5. Output $(\Pi_*,\Pi_{\mathsf{mem}},\Pi_{\mathbb{i}})$.
- $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}}}(\mathbb{x})$: Accept if and only if all of the following test accept.
  1. *Membership test*: $\mathbf{V}_{\mathsf{PCPP}}^{\Pi_*,\tilde{\Pi}_{\mathsf{mem}}}(M_*,\mathbb{x},T_*) = 1$.
  2. *Consistency test*: Parse $\tilde{\Pi}_{\mathbb{i}} = \{\tilde{\Pi}_i\}_{i\in[\mathsf{k}]}$. Choose PCPP verifier randomness α. For every $i \in [\mathsf{k}]$ test that $\mathbf{V}_{\mathsf{PCPP}}^{(\tilde{\pi}_i,\tilde{\Pi}_*),\tilde{\Pi}_i}(M_i,\perp,T_i;\alpha) = 1$. (Using the verifier of the oblivious PCPP for $\mathcal{M}$).
- $\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_i)$: $\mathbf{Dec}(\tilde{\pi}_i)$.
- $\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})$: Let $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}})$ be the codeword closest to $\tilde{\Pi}_*$. Output $\tilde{\mathbb{w}}$.

Proof. We prove completeness, then decodability, and finally analyze complexity measures.

Completeness. Fix $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. We show that both of $\mathbf{V}_{\mathsf{PCP}}$'s tests pass with probability 1 and therefore $\mathbf{V}_{\mathsf{PCP}}$ always accepts. Let $\pi_1, \ldots, \pi_{\mathsf{k}}$ and $\Pi_*, \Pi_{\mathsf{mem}}, \Pi_{\mathbb{i}}$ be the proofs generated by the indexer and prover respectively where $\Pi_{\mathbb{i}} := \{\Pi_i\}_{i \in [\mathsf{k}]}$.

1. *Membership test*: Since $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\Pi_* = \mathbf{Enc}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{w})$ we have that $M_*(\mathbb{x}, \Pi_*) = 1$. By the perfect completeness of the PCP of proximity, since $\Pi_{\mathsf{mem}} = \mathbf{P}_{\mathsf{PCPP}}(M_*, \mathbb{x}, T_*, \Pi_*)$, we have that
$$\Pr\left[\ \mathbf{V}_{\mathsf{PCPP}}^{\Pi_*, \Pi_{\mathsf{mem}}}(M_*, \mathbb{x}, T_*) = 1\ \right] = 1\,.$$
2. *Consistency test*: Since $\Pi_* = \mathbf{Enc}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{w})$ and for every $i \in [\mathsf{k}]$, $\pi_i = \mathbf{Enc}(\mathbb{i}[i])$, we have that for every i, $M_i(\pi_i, \Pi_*) = 1$. Hence, since
$$\Pi_i = \mathbf{P}_{\mathsf{PCPP}}(M_i, \perp, T_i, (\pi_i, \Pi_*))\,,$$
by the perfect completeness of the PCP of proximity:
$$\Pr\left[\ \mathbf{V}_{\mathsf{PCPP}}^{(\pi_i, \Pi_*), \Pi_i}(M_i, \perp, T_i) = 1\ \right] = 1\,.$$

Decodability. Fix $\mathbb{x}$, $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}$ and $\tilde{\Pi}_{\mathbb{i}} = \{\tilde{\Pi}_i\}_{i \in [\mathsf{k}]}$. Suppose that:
$$\Pr\left[\ \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}, \tilde{\Pi}_{\mathbb{i}}}(\mathbb{x}) = 1\ \right] > \beta_{\mathsf{PCPP}}\,.$$
We show that $(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}, \tilde{\Pi}_{\mathbb{i}})) \in \mathcal{R}$. To do so we give two claims, each relating to a different test done by the verifier. The first says that the verifier's membership test implies that $\tilde{\Pi}_*$ encodes strings that put $\mathbb{x}$ in $\mathcal{R}$.

Claim 3.7.9. *There exist* $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ *and* $\tilde{\mathbb{w}}$ *such that:*
1. Valid encoding*:* $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \tilde{\mathbb{w}}) = \mathbf{Dec}(\tilde{\Pi}_*)$.
2. Membership*:* $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \mathbb{x}, \tilde{\mathbb{w}}) \in \mathcal{R}$.

Proof. We show that there exist $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that $\Delta(\widehat{\Pi}_*, \tilde{\Pi}_*) \leq \delta_{\mathsf{PCPP}}$ where $\widehat{\Pi}_* := \mathbf{Enc}(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \tilde{\mathbb{w}})$ and which place $\mathbb{x}$ in the relation. This will imply the claim since the distance of the error-correcting code is at most $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{ECC}}$, and so $\widehat{\Pi}_*$ and $\tilde{\Pi}_*$ decode to the same value.

Suppose towards contradiction that for every $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that $\Delta(\widehat{\Pi}_*, \tilde{\Pi}_*) \leq \delta_{\mathsf{PCPP}}$ (where $\widehat{\Pi}_*$ is defined as before) we have that $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \mathbb{x}, \tilde{\mathbb{w}}) \notin \mathcal{R}$. This means that $\tilde{\Pi}_*$ has distance greater than δ_{PCPP} from every $\widehat{\Pi}_*$ such that $M_*(\mathbb{x}, \widehat{\Pi}_*) = 1$. Hence by soundness of the PCPP system:
$$\Pr\left[\ \mathbf{V}_{\mathsf{PCPP}}^{\tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}}(M_*, \mathbb{x}, T_*) = 1\ \right] \leq \beta_{\mathsf{PCPP}}\,.$$

$\mathbf{V}_{\mathsf{PCP}}$ runs this test in Item 1 and hence will accept with probability at most β_{PCPP} in contradiction to the assumption that

$$\Pr\left[\ \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}}}(\mathbb{x}) = 1\ \right] > \beta_{\mathsf{PCPP}}\,.$$

□

We now show that the indexer proofs $\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}}$ must be consistent with the encoding $\tilde{\Pi}_*$.

Claim 3.7.10. *There exist* $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ *and* $\tilde{\mathbb{w}}$ *such that:*
1. Valid encoding*:* $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}}) = \mathbf{Dec}(\tilde{\Pi}_*)$.
2. Consistency*: For every* $i \in [\mathsf{k}]$, $\tilde{\mathbb{i}}[i] := \mathbf{Dec}(\tilde{\pi}_i)$.

Proof. We show that there exist $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that $\Delta(\widehat{\Pi}_*,\tilde{\Pi}_*) \le \delta_{\mathsf{PCPP}}$ where $\widehat{\Pi}_* := \mathbf{Enc}(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}})$. Additionally we have that for every $i \in [\mathsf{k}]$, $\Delta(\widehat{\pi}_i,\tilde{\pi}_i) \le \delta_{\mathsf{PCPP}}$ where $\widehat{\pi}_i := \mathbf{Enc}(\tilde{\mathbb{i}}[i])$. This will imply the claim since the distance of the error-correcting code is at most $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{ECC}}$, and so $\widehat{\Pi}_*$ and $\tilde{\Pi}_*$ decode to the same value. Similarly $\widehat{\pi}_i$ and $\tilde{\pi}_i$ decode to the same value.

Suppose towards contradiction that there exists $i \in [\mathsf{k}]$ such that for every pair $\widehat{\pi}_i$ and $\widehat{\Pi}_*$ such that $M_i(\bot,\widehat{\pi}_i,\widehat{\Pi}_*) = 1$ (which implies that they have the required consistency), at least one of the following holds: (i) $\Delta(\widehat{\pi}_i,\tilde{\pi}_i) > \delta_{\mathsf{PCPP}}$, (ii) $\Delta(\widehat{\Pi}_*,\tilde{\Pi}_*) > \delta_{\mathsf{PCPP}}$. Then by the soundness property of the *multi-input* PCPP system:

$$\Pr\left[\ \mathbf{V}_{\mathsf{PCPP}}^{(\tilde{\pi}_i,\tilde{\Pi}_*),\tilde{\Pi}_i}(M_i,\bot,T_i) = 1\ \right] \le \beta_{\mathsf{PCPP}}\,,$$

in contradiction to the assumption that $\mathbf{V}_{\mathsf{PCP}}$, that runs the above test in Item 2, accepts with probability greater than β_{PCPP}. □

We now prove decodability. Under the assumption that the verifier accepts with probability greater than β_{PCPP}, by Theorem 3.7.9 and Theorem 3.7.10, there exist $\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that:
1. $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\mathbb{x},\tilde{\mathbb{w}}) \in \mathcal{R}$.
2. $(\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\tilde{\mathbb{w}}) = \mathbf{Dec}(\tilde{\Pi}_*)$.
3. For every $i \in [\mathsf{k}]$: $\tilde{\mathbb{i}}[i] = \mathbf{Dec}(\tilde{\pi}_i)$.
Putting the above items together with the definition of the decoders we have that:

$$\begin{aligned}(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1),\ldots,\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}),\mathbb{x},\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})) &= (\mathbf{Dec}(\tilde{\pi}_1),\ldots,\mathbf{Dec}(\tilde{\pi}_{\mathsf{k}}),\mathbb{x},\tilde{\mathbb{w}})\\ &= (\tilde{\mathbb{i}}[1],\ldots,\tilde{\mathbb{i}}[\mathsf{k}],\mathbb{x},\tilde{\mathbb{w}}) \in \mathcal{R}\,.\end{aligned}$$

Efficiency. We analyze the efficiency parameters of the PCP:

- *Indexer alphabet.* The indexer alphabet size is 2.
- *Prover alphabet.* The prover writes its proof in groups of k bits. The alphabet size is 2^{k}.

- *Indexer proof length.* $\mathbf{I}_{\mathsf{PCP}}$ uses **Enc** on a bit-string of length $|\mathbb{i}[i]|$, so the proof length is $N_{\mathsf{ECC}}(|\mathbb{i}[i]|)$.
- *Prover proof length.* $\mathbf{P}_{\mathsf{PCP}}$ outputs the encoding of the string $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{w}$, and outputs $\mathsf{k}+1$ proofs for the PCP of proximity. The proofs in the consistency test part are interleaved into symbols. Thus the proof has length $2 \cdot \mathsf{l}_{\mathsf{PCPP}} + N_{\mathsf{ECC}}(|\mathbb{w}| + \sum_{i=1}^{\mathsf{k}} |\mathbb{i}[i]|)$.
- *Query complexity.* $\mathbf{V}_{\mathsf{PCP}}$ makes $\mathsf{q}_{\mathsf{PCPP}}$ queries to each of the indexer proofs in the consistency tests. The consistency check is done k times with the same randomness, and the same family of machines $\mathcal{M}$. The PCPP system is oblivious, and so all of these PCPPs make queries to exactly the same locations – which are bundled together into one symbol by the prover. Thus this test makes only $\mathsf{q}_{\mathsf{PCPP}}$ queries. The verifier additionally makes $\mathsf{q}_{\mathsf{PCPP}}$ queries to the prover proof in the membership test – a total of $2 \cdot \mathsf{q}_{\mathsf{PCPP}}$.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{PCP}}$ runs the membership test with randomness $\mathsf{r}_{\mathsf{PCPP}}$, chooses new PCPP randomness and runs each of the consistency checks *with the same randomness*. Therefore it uses $2 \cdot \mathsf{r}_{\mathsf{PCPP}}$ random bits.
- *Indexer running time.* $\mathbf{I}_{\mathsf{PCP}}$ encodes $\mathbb{i}[i]$ in time $\mathsf{et}_{\mathsf{ECC}}(|\mathbb{i}[i]|)$.
- *Prover running time.* $\mathbf{P}_{\mathsf{PCP}}$ encodes a string of length $|\mathbb{w}| + \sum_{i=1}^{\mathsf{k}} |\mathbb{i}[i]|$ in time $\mathsf{et}_{\mathsf{ECC}}(|\mathbb{w}| + \sum_{i=1}^{\mathsf{k}} |\mathbb{i}[i]|)$ and computes $\mathsf{k}+1$ PCP of proximity proofs, each in time $\mathsf{pt}_{\mathsf{PCPP}}$. All together time $\mathsf{et}_{\mathsf{ECC}}(|\mathbb{w}| + \sum_{i=1}^{\mathsf{k}} |\mathbb{i}[i]|) + (\mathsf{k}+1) \cdot \mathsf{pt}_{\mathsf{PCPP}}$.
- *Verifier running time.* $\mathbf{V}_{\mathsf{PCP}}$ runs the PCPP verifier $\mathsf{k}+1$ times, taking time $(\mathsf{k}+1) \cdot \mathsf{vt}_{\mathsf{PCPP}}$.
- *Decoder running time.* The running time of $\mathbf{iD}_{\mathsf{PCP}}$ is $\mathsf{dt}_{\mathsf{ECC}}(|\mathbb{i}[i]|)$. The runnign time of $\mathbf{wD}_{\mathsf{PCP}}$ is $\mathsf{dt}_{\mathsf{ECC}}(|\mathbb{w}| + \sum_{i=1}^{\mathsf{k}} |\mathbb{i}[i]|)$.
- *Adaptivity.* If $\mathbf{V}_{\mathsf{PCPP}}$ is non-adaptive then so is $\mathbf{V}_{\mathsf{PCP}}$.

□

3.8 ID-PCPs with constant query complexity over a binary alphabet

We construct index-decodable PCPs that make $O(1)$ queries to every oracle, over the binary alphabet. We begin in Section 3.8.1 by showing that proof composition preserves index-decodability when the outer index-decodable PCP is prover-robust. Then, in Section 3.8.2, we show how to transform the index-decodable PCP constructed in Section 3.7 into a prover-robust index-decodable PCP. Combining these, we prove the following theorem.

Theorem 3.8.1 (restatement of Theorem 3)**.** *Let $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ be a multi-indexed relation decidable in* NTIME(T)*. Then $\mathcal{R}$ has a non-adaptive index-decodable PCP* $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ *with the parameters below.*

Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length (per proof)	$O(\lvert\mathbb{i}[i]\rvert)$
Prover proof length	$\mathrm{poly}(T)$
Alphabet size	2
Queries per oracle	$O(1)$
Randomness	$O(\log T)$
Decodability bound	$O(1)$
Indexer running time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Prover running time	$\mathrm{poly}(T)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \log T)$
Index decoding time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Witness decoding time	$\mathrm{poly}(T)$

Proof. We take the robust index-decodable PCP of Theorem 3.8.4 and compose it using Theorem 3.8.2 with the PCP of proximity achieved by Theorem 2.1.1. □

3.8.1 Proof composition preserves index-decodability

We show that, in proof composition of PCPs [AS98], if the outer PCP of the proof is index-decodable then the composed PCP is also index-decodable (given the outer index-decodable PCP has good enough prover-robustness).

Theorem 3.8.2. *Let $\mathsf{PCP}_{\mathsf{out}} = (\mathbf{I}_{\mathsf{out}}, \mathbf{P}_{\mathsf{out}}, (\mathbf{V}^{\mathsf{qry}}_{\mathsf{out}}, \mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}), \mathbf{iD}_{\mathsf{out}}, \mathbf{wD}_{\mathsf{out}})$ be a non-adaptive index-decodable PCP for a relation $\mathcal{R}$ with prover-robustness σ_{out} and $\mathsf{PCP}_{\mathsf{in}} = (\mathbf{P}_{\mathsf{in}}, \mathbf{V}_{\mathsf{in}})$ be a non-adaptive PCP of proximity for* NP *with proximity $\delta_{\mathsf{in}} \leq \sigma_{\mathsf{out}}$. Then Theorem 3.8.3 is a non-adaptive index-decodable PCP $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ for $\mathcal{R}$ with the parameters below.*

Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length	$\mathsf{l}_{\mathsf{out},\mathbf{I}}$
Proof length	$\mathsf{l}_{\mathsf{out},\mathbf{P}}$
Alphabet size	λ_{out}
Queries to indexer proof	$\mathsf{iq}_{\mathsf{out}}$
Queries to prover proof	$\mathsf{pq}_{\mathsf{out}}$
Randomness	$\mathsf{r}_{\mathsf{out}}$
Prover-robustness	σ_{out}
Decodability bound	κ_{out}
Indexer running time	$\mathsf{it}_{\mathsf{out}}$
Prover running time	$\mathsf{pt}_{\mathsf{out}}$
Verifier running time	$\mathsf{vt}_{\mathsf{out}}$
Index decoding time	$\mathsf{idt}_{\mathsf{out}}$
Witness decoding time	$\mathsf{wdt}_{\mathsf{out}}$

+

PCPP for $\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}$	
Proof length	l_{in}
Alphabet size	λ_{in}
Queries	q_{in}
Randomness	r_{in}
Proximity	δ_{in}
Soundness error	β_{in}
Prover running time	$\mathsf{pt}_{\mathsf{in}}$
Verifier running time	$\mathsf{vt}_{\mathsf{in}}$

$\longrightarrow$

Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length	$\mathsf{l}_{\mathsf{out},\mathbf{I}}$
Prover proof length	$\mathsf{l}_{\mathsf{out},\mathbf{P}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot \mathsf{l}_{\mathsf{in}}$
Alphabet size	$\max\{\lambda_{\mathsf{out}}, \lambda_{\mathsf{in}}\}$
Queries to indexer proof	$\mathsf{iq}_{\mathsf{out}}$
Queries to prover proof	q_{in}
Randomness	$\mathsf{r}_{\mathsf{out}} + \mathsf{r}_{\mathsf{in}}$
Decodability bound	$\kappa_{\mathsf{out}} + (1 - \kappa_{\mathsf{out}}) \cdot \beta_{\mathsf{in}}$
Indexer running time	$\mathsf{it}_{\mathsf{out}}$
Prover running time	$\mathsf{pt}_{\mathsf{out}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot (\mathsf{vt}_{\mathsf{out}} + \mathsf{pt}_{\mathsf{in}})$
Verifier running time	$\mathsf{vt}_{\mathsf{out}} + \mathsf{vt}_{\mathsf{in}}$
Index decoding time	$\mathsf{idt}_{\mathsf{out}}$
Witness decoding time	$\mathsf{wdt}_{\mathsf{out}}$

Construction 3.8.3. We construct the index-decodable PCP $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ below.

- $\mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$: Output $\pi_i := \mathbf{I}_{\mathsf{out}}(\mathbb{i}[i])$.
- $\mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$:
  1. For every $i \in [\mathsf{k}]$, compute the indexer proof $\pi_i := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$.
  2. Compute the prover proof $\Pi_{\mathsf{out}} := \mathbf{P}_{\mathsf{out}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$ and set $\mathbf{\Pi}_{\mathbb{i}} := (\pi_1, \ldots, \pi_{\mathsf{k}})$.
  3. For every $\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}$, compute $(Q_{\mathbb{i}}, Q_*) := \mathbf{V}^{\mathsf{qry}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}})$ and set $\mathbb{x}_{\mathsf{in}} := \Pi[Q_*]$. Compute the PCPP string $\Pi_{\mathsf{in}}[\alpha_{\mathsf{out}}] := \mathbf{P}_{\mathsf{in}}\big(\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}, (\mathbb{x}, \alpha_{\mathsf{out}}, \mathbf{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}]), \mathsf{vt}_{\mathsf{out}}, \mathbb{x}_{\mathsf{in}}\big)$ (i.e., generate a proof that the machine $M_{\mathsf{in}} := \mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}}, \mathbf{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}], \cdot)$ accepts $\mathbb{x}_{\mathsf{in}}$).
  4. Output $\Pi := (\Pi_{\mathsf{out}}, \Pi_{\mathsf{in}})$ where $\Pi_{\mathsf{in}} := \{\Pi_{\mathsf{in}}[\alpha_{\mathsf{out}}]\}_{\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}}$.
- $\mathbf{V}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}}_{\mathsf{PCP}}(\mathbb{x})$:
  1. Parse $\tilde{\Pi} = (\tilde{\Pi}_{\mathsf{out}}, \{\tilde{\Pi}_{\mathsf{in}}[\alpha_{\mathsf{out}}]\}_{\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}})$ and set $\tilde{\mathbf{\Pi}}_{\mathbb{i}} := (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$ for convenience.

2. Sample randomness $\alpha_{\mathsf{out}} \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{out}}}$ and compute the query sets $(Q_{\mathbb{i}}, Q_*) := \mathbf{V}^{\mathsf{qry}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}})$.
3. Check that $\mathbf{V}^{\tilde{\Pi}_{\mathsf{out}}[Q_*], \tilde{\Pi}_{\mathsf{in}}[\alpha_{\mathsf{out}}]}_{\mathsf{in}}\big(\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}, (\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}]), \mathsf{vt}_{\mathsf{out}}\big) = 1$.

- $\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_j)$: output $\mathbf{iD}_{\mathsf{out}}(\tilde{\pi}_j)$.
- $\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_{\mathsf{out}}, \tilde{\Pi}_{\mathsf{in}})$: output $\mathbf{wD}_{\mathsf{out}}(\tilde{\Pi}_{\mathsf{out}})$.

Proof of Theorem 3.8.2. First we argue completeness, then argue decodability, and, finally, analyze efficiency measures of the resulting PCP.

Completeness. Fix $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and let $\boldsymbol{\Pi}_{\mathbb{i}} = (\pi_1, \ldots, \pi_{\mathsf{k}})$, Π_{out} and $\{\Pi_{\mathsf{in}}[\alpha_{\mathsf{out}}]\}_{\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}}$ be the proofs output by the honest indexer $\mathbf{I}_{\mathsf{PCP}}$ and prover $\mathbf{P}_{\mathsf{PCP}}$. By the (perfect) completeness of the outer PCP,

$$\Pr\left[\, \mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}], \Pi_{\mathsf{out}}[Q_*]) = 1 \mid (Q_{\mathbb{i}}, Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x}, \alpha) \,\right] = 1\,.$$

Hence, for every $\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}$, by the (perfect) completeness of the inner PCP (of proximity), for $\Pi_{\mathsf{in}}[\alpha_{\mathsf{out}}]$ output by $\mathbf{P}_{\mathsf{in}}$ given $\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}$, $(\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}])$, $\mathsf{vt}_{\mathsf{out}}$ (which upper bounds the running time of $\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}$) and $\Pi_{\mathsf{out}}[Q_*]$, it holds that

$$\Pr_{\alpha_{\mathsf{in}}}\left[\, \mathbf{V}^{\Pi_{\mathsf{out}}[Q_*], \Pi_{\mathsf{in}}[\alpha_{\mathsf{out}}]}_{\mathsf{in}}\big(\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}, (\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}]), \mathsf{vt}_{\mathsf{out}}; \alpha_{\mathsf{in}}\big) = 1 \,\right] = 1\,.$$

We conclude that the composed PCP also has perfect completeness:

$$\Pr\left[\, \mathbf{V}^{\pi_1, \ldots, \pi_{\mathsf{k}}, \Pi}_{\mathsf{PCP}}(\mathbb{x}) = 1 \,\right] = 1\,.$$

Decodability. Fix $\mathbb{x}$ and malicious proofs $\tilde{\boldsymbol{\Pi}}_{\mathbb{i}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$ and $\tilde{\Pi} = (\tilde{\Pi}_{\mathsf{out}}, \{\tilde{\Pi}_{\mathsf{in}}[\alpha_{\mathsf{out}}]\})$. Suppose that:

$$\Pr\left[\, \mathbf{V}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}}_{\mathsf{PCP}}(\mathbb{x}) = 1 \,\right] > \kappa_{\mathsf{out}} + (1 - \kappa_{\mathsf{out}}) \cdot \beta_{\mathsf{in}}\,.$$

For every choice of randomness $\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}$ let $(Q_{\mathbb{i}}, Q_*) := \mathbf{V}^{\mathsf{qry}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}})$ and set $\tilde{A}_{\alpha_{\mathsf{out}}} := \tilde{\Pi}_{\mathsf{out}}[Q_*]$. We consider the two possibilities for $\tilde{A}_{\alpha_{\mathsf{out}}}$.

1. There exists some $\tilde{A}'$ such that $\Delta(\tilde{A}', \tilde{A}_{\alpha_{\mathsf{out}}}) \leq \sigma_{\mathsf{out}}$ and $\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}], \tilde{A}') = 1$. In this case, we cannot rule out that $\mathbf{V}_{\mathsf{in}}$ accepts with high probability. So we can trivially write

$$\Pr_{\alpha_{\mathsf{in}}}\left[\, \mathbf{V}^{\tilde{A}_{\alpha_{\mathsf{out}}}, \tilde{\Pi}_{\mathsf{in}}[\alpha_{\mathsf{out}}]}_{\mathsf{in}}\big(\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}, (\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}]), \mathsf{vt}_{\mathsf{out}}; \alpha_{\mathsf{in}}\big) = 1 \,\right] \leq 1\,.$$

2. There *does not* exist a set $\tilde{A}'$ such that $\Delta(\tilde{A}', \tilde{A}_{\alpha_{\mathsf{out}}}) \leq \sigma_{\mathsf{out}}$ and $\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}(\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}], \tilde{A}') = 1$. Since $\delta_{\mathsf{in}} \leq \sigma_{\mathsf{out}}$, $\tilde{A}_{\alpha_{\mathsf{out}}}$ is far enough from any true claim that the proximity soundness property of $\mathbf{V}_{\mathsf{in}}$ applies:

$$\Pr_{\alpha_{\mathsf{in}}}\left[\, \mathbf{V}^{\tilde{A}_{\alpha_{\mathsf{out}}}, \tilde{\Pi}_{\mathsf{in}}[\alpha_{\mathsf{out}}]}_{\mathsf{in}}\big(\mathbf{V}^{\mathsf{dc}}_{\mathsf{out}}, (\mathbb{x}, \alpha_{\mathsf{out}}, \boldsymbol{\Pi}_{\mathbb{i}}[Q_{\mathbb{i}}]), \mathsf{vt}_{\mathsf{out}}; \alpha_{\mathsf{in}}\big) = 1 \,\right] \leq \beta_{\mathsf{in}}\,.$$

Hence, letting p be the probability that α_{out} induces a choice of queries whose answers $\tilde{A}_{\alpha_{\mathsf{out}}}$ are such that Item 1 occurs, we have that $\mathbf{V}_{\mathsf{PCP}}$ accepts with probability at most $p + (1-p) \cdot \beta_{\mathsf{in}}$. By assumption, the probability that $\mathbf{V}_{\mathsf{PCP}}$ accepts is greater than $\kappa_{\mathsf{out}} + (1-\kappa_{\mathsf{out}}) \cdot \beta_{\mathsf{in}}$. From this we can infer that $p > \kappa_{\mathsf{out}}$. By the prover-robust decodability of the outer (index-decodable) PCP, we deduce that:

$$(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})) = (\mathbf{iD}_{\mathsf{out}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{out}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{out}}(\tilde{\Pi}_{\mathsf{out}})) \in \mathcal{R}\,.$$

Efficiency. We analyze the efficiency parameters of the resulting PCP.

- *Alphabet.* The new PCP involves the alphabet of the outer index-decodable PCP, which has size λ_{out}, and the alphabet of the inner PCP of proximity, which has size λ_{in}. One can use the same alphabet to write both, in which case its size would be $\max\{\lambda_{\mathsf{out}}, \lambda_{\mathsf{in}}\}$.
- *Indexer proof length.* $\mathbf{I}_{\mathsf{PCP}}$ outputs the same indexer proofs as $\mathbf{I}_{\mathsf{out}}$, which are of length $\mathsf{l}_{\mathsf{out},\mathbf{I}}$.
- *Prover proof length.* $\mathbf{P}_{\mathsf{PCP}}$ sends the proof (of length $\mathsf{l}_{\mathsf{out},\mathbf{P}}$) of the index-decodable PCP and also, for every $\alpha_{\mathsf{out}} \in \{0,1\}^{\mathsf{r}_{\mathsf{out}}}$, sends a proof (of length l_{in}) for the inner PCP for randomness α_{out}. Hence the total proof length is $\mathsf{l}_{\mathsf{out},\mathbf{P}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot \mathsf{l}_{\mathsf{in}}$.
- *Query complexity.* $\mathbf{V}_{\mathsf{PCP}}$ makes as many queries as $\mathbf{V}_{\mathsf{in}}$, which is q_{in}.
- *Randomness complexity.* $\mathbf{V}_{\mathsf{PCP}}$ samples randomness for $\mathbf{V}_{\mathsf{out}}$ and $\mathbf{V}_{\mathsf{in}}$, using $\mathsf{r}_{\mathsf{out}} + \mathsf{r}_{\mathsf{in}}$ random bits.
- *Indexer running time.* $\mathbf{I}_{\mathsf{PCP}}$ runs runs $\mathbf{I}_{\mathsf{out}}$, and so its running time is $\mathsf{it}_{\mathsf{out}}$.
- *Prover running time.* $\mathbf{P}_{\mathsf{PCP}}$ runs $\mathbf{P}_{\mathsf{out}}$ once and $\mathbf{V}_{\mathsf{out}}^{\mathsf{qry}}, \mathbf{P}_{\mathsf{in}}$ a total of $2^{\mathsf{r}_{\mathsf{out}}}$ times, and so its running time is $\mathsf{pt}_{\mathsf{out}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot (\mathsf{vt}_{\mathsf{out}} + \mathsf{pt}_{\mathsf{in}})$.
- *Verifier running time.* $\mathbf{V}_{\mathsf{PCP}}$ runs $\mathbf{V}_{\mathsf{out}}^{\mathsf{qry}}$ to compute its query locations and runs $\mathbf{V}_{\mathsf{in}}$ to decide, thereby running in time at most $\mathsf{vt}_{\mathsf{out}} + \mathsf{vt}_{\mathsf{in}}$.
- *Decoder running time.* The decoders have the same running time as the outer PCP decoders.

□

3.8.2 Robustification

We now show how to get a prover-robust index-decodable PCP with a binary alphabet and large number of queries to the prover proof. This is later reduced by using proof composition.

Theorem 3.8.4. *Let $\mathcal{R} = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ be a multi-indexed relation decidable in* NTIME(T). *Then $\mathcal{R}$ a non-adaptive prover-robust index-decodable PCP with the following parameters:*

Prover-robust Index-Decodable PCP for $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in \mathcal{R}$	
Indexer proof length (per proof)	$O(\lvert\mathbb{i}[i]\rvert)$
Proof length	poly(T)
Alphabet size	2
Queries to indexer proof	$O(1)$
Queries to prover proof	$O(\mathsf{k})$
Randomness	$O(\log T)$
Prover-robustness	$\Omega(1)$
Decodability bound	$O(1)$
Indexer running time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Prover running time	poly(T)
Verifier running time	poly$(\mathbb{x}, \mathsf{k}, T)$
Index decoding time	$\tilde{O}(\lvert\mathbb{i}[i]\rvert)$
Witness decoding time	poly(T)

We first describe the construction, and then prove Theorem 3.8.4.

Construction 3.8.5. Let $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, (\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}), \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be a non-adaptive index-decodable PCP for a relation $\mathcal{R}$ and $\mathsf{ECC} = (\mathbf{Enc}, \mathbf{Dec})$ be a $(r, \delta_{\mathsf{ECC}})$-code with $r(k) = c \cdot k$ for constant c. Let $\mathsf{l}_{\mathsf{PCP},\mathbf{P}}$ and $\Sigma_{\mathbf{P}}$ be the prover proof length and alphabet respectively.

- $\mathbf{I}(\mathbb{i}[i])$: Output $\pi_i := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$.
- $\mathbf{P}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$: Compute $\Pi' := \mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$ and output $\Pi := \mathbf{Enc}(\Pi', \mathsf{l}_{\mathsf{PCP},\mathbf{P}})$.
- $\mathbf{V}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}}(\mathbb{x})$:
  1. Sample randomness α and generate $(Q_{\mathbb{i}}, Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x}, \alpha)$ the queries the PCP verifier makes given instance $\mathbb{x}$ and randomness α. $Q_{\mathbb{i}}$ are the queries made to the indexer proofs and Q_* are the queries made to the prover proof.
  2. Let $A := \{\tilde{\Pi}[q] \mid q \in Q_*\}$ and $A_{\mathsf{PCP}} := \{\mathbf{Dec}(a) \mid a \in A\}$. Let $\tilde{\mathbf{\Pi}}_{\mathbb{i}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$. Accept if and only if $\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x}, \alpha, \tilde{\mathbf{\Pi}}_{\mathbb{i}}[Q_{\mathbb{i}}], A_{\mathsf{PCP}}) = 1$.
- $\mathbf{iD}(\tilde{\pi}_i)$: output $\mathbf{iD}_{\mathsf{PCP}}(\tilde{\Pi})$.
- $\mathbf{wD}(\tilde{\Pi})$: output $\mathbf{wD}_{\mathsf{PCP}}(\mathbf{Dec}(\tilde{\Pi}, \mathsf{l}_{\mathsf{PCP},\mathbf{P}}))$.

Proof of Theorem 3.8.4. We use Theorem 3.8.5 where the index-decodable PCP used is the one of Theorem 3.7.1. We use an error correcting code with parameters as in Theorem 2.2.1. We first argue completeness, then decodability and, finally, we analyze the other complexity measures of the resulting PCP.

Completeness. Fix $(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{x},\mathbb{w}) \in \mathcal{R}$. Let $\pi_1,\ldots,\pi_\mathsf{k}$ and Π be the proofs generated by the indexer and the prover respectively. Since the prover is honest, $\mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{x},\mathbb{w}) = \mathbf{Dec}(\Pi, \mathsf{I}_{\mathsf{PCP},\mathbf{P}})$. Hence, letting $\Pi' := \mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1],\ldots,\mathbb{i}[\mathsf{k}],\mathbb{x},\mathbb{w})$ and $\tilde{\mathbf{\Pi}}_\mathbb{i} = (\tilde{\pi}_1,\ldots,\tilde{\pi}_\mathsf{k})$, we have:

$$\begin{aligned}
\Pr\left[\, \mathbf{V}^{\pi_1,\ldots,\pi_\mathsf{k},\Pi}(\mathbb{x}) = 1 \,\right] &= \Pr_\alpha\left[\, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}], A_{\mathsf{PCP}}) = 1 \;\middle|\; \begin{array}{r}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x},\alpha)\\ A_{\mathsf{PCP}} := \{\, \mathbf{Dec}(\Pi[q]) \mid q \in Q_* \,\}\end{array} \,\right] \\
&= \Pr_\alpha\left[\, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}], A_{\mathsf{PCP}}) = 1 \;\middle|\; \begin{array}{r}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x},\alpha)\\ A_{\mathsf{PCP}} := \{\, \Pi'[q] \mid q \in Q_* \,\}\end{array} \,\right] \\
&= \Pr\left[\, \mathbf{V}^{\pi_1,\ldots,\pi_\mathsf{k},\Pi'}_{\mathsf{PCP}}(\mathbb{x}) = 1 \,\right] \\
&= 1\,.
\end{aligned}$$

Prover-robust decodability. Fix an instance $\mathbb{x}$ and proofs $\tilde{\mathbf{\Pi}}_\mathbb{i} = (\tilde{\pi}_1,\ldots,\tilde{\pi}_\mathsf{k})$ and $\tilde{\Pi}$. Denote by $\kappa_{\mathsf{PCP}} = O(1)$, and $\mathsf{q}_{\mathsf{PCP}} = O(1)$ the decodability bound and number of queries to the prover proof respectively. Let $\delta_{\mathsf{ECC}} = \Omega(1)$ be the distance of the error correcting code such that $\frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}} = \Omega(1)$. Denote by $\mathbf{V}^{\mathsf{qry}}$ and $\mathbf{V}^{\mathsf{dc}}$ the query generation and decision predicate of $\mathbf{V}$ respectively. Suppose that

$$\Pr_\alpha\left[\, \exists\, A' \text{ s.t. } \begin{array}{c}\mathbf{V}^{\mathsf{dc}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],A') = 1\\ \wedge\ \Delta(A',A) \le \frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}}\end{array} \;\middle|\; \begin{array}{c}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}(\mathbb{x},\alpha)\\ A := \{\, \tilde{\Pi}[q] \mid q \in Q_* \,\}\end{array} \,\right] > \kappa_{\mathsf{PCP}}\,.$$

We show that this implies that:

$$(\mathbf{iD}(\tilde{\pi}_1),\ldots,\mathbf{iD}(\tilde{\pi}_\mathsf{k}),\mathbb{x},\mathbf{wD}(\tilde{\Pi})) \in \mathcal{R}\,.$$

Notice that

$$\Pr_\alpha\left[\, \exists\, A' \text{ s.t. } \begin{array}{c}\mathbf{V}^{\mathsf{dc}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],A') = 1\\ \wedge\ \Delta(A',A) \le \frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}}\end{array} \;\middle|\; \begin{array}{c}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}(\mathbb{x},\alpha)\\ A := \{\, \tilde{\Pi}[q] \mid q \in Q_* \,\}\end{array} \,\right],$$

is equal to

$$\Pr_\alpha\left[\, \exists\, A' \text{ s.t. } \begin{array}{c}\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}\left(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],\{\mathbf{Dec}(a) \mid a \in A'\}\right) = 1\\ \wedge\ \Delta(A',A) \le \frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}}\end{array} \;\middle|\; \begin{array}{c}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x},\alpha)\\ A := \{\, \tilde{\Pi}[q] \mid q \in Q_* \,\}\end{array} \,\right].$$

Fix randomness α. Let $(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x},\alpha)$ and $A = \{\, \mathbf{\Pi'}[q] \mid q \in Q_* \,\}$. Suppose that A' is the set closest to A such that $\mathbf{V}^{\mathsf{dc}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],A') = 1$ and that $\Delta(A',A) \le \frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}}$. By a simple counting argument it must be that for every i, $\Delta(A'[i],A[i]) \le \frac{\delta_{\mathsf{ECC}}}{4} < \frac{\delta_{\mathsf{ECC}}}{2}$, and so $\mathbf{Dec}(A[i]) = \mathbf{Dec}(A'[i])$. Moreover, since $A[i] = \tilde{\Pi}[Q_*[i]]$, it follows that $\mathbf{Dec}(A'[i]) = \mathbf{Dec}(\tilde{\Pi}[Q[{}_*i]])$. Hence, considering the decoded proof $\tilde{\Pi}' = \mathbf{Dec}(\tilde{\Pi}, \mathsf{I}_{\mathsf{PCP},\mathbf{P}})$, we have:

$$\begin{aligned}
&\Pr_\alpha\left[\, \exists\, A' \text{ s.t. } \begin{array}{c}\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}\left(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],\{\mathbf{Dec}(a) \mid a \in A'\}\right) = 1\\ \wedge\ \Delta(A',A) \le \frac{\delta_{\mathsf{ECC}}}{4\mathsf{q}_{\mathsf{PCP}}}\end{array} \;\middle|\; \begin{array}{c}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}(\mathbb{x},\alpha)\\ A := \{\, \tilde{\Pi}[q] \mid q \in Q_* \,\}\end{array} \,\right] \\
&\le \Pr_\alpha\left[\, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_\mathbb{i}[Q_\mathbb{i}],A_{\mathsf{PCP}}) = 1 \;\middle|\; \begin{array}{c}(Q_\mathbb{i},Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}(\mathbb{x},\alpha)\\ A_{\mathsf{PCP}} = \{\, \tilde{\Pi}'[q] \mid q \in Q_* \,\}\end{array} \,\right],
\end{aligned}$$

and so

$$\Pr\left[\ \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}'}(\mathbb{x})\ \right] = \Pr_{\alpha}\left[\mathbf{V}_{\mathsf{PCP}}^{\mathsf{dc}}(\mathbb{x},\alpha,\tilde{\mathbf{\Pi}}_{\mathbb{i}}[Q_{\mathbb{i}}],A_{\mathsf{PCP}})=1 \;\middle|\; \begin{array}{c}(Q_{\mathbb{i}},Q_*)\leftarrow\mathbf{V}_{\mathsf{PCP}}^{\mathsf{qry}}(\mathbb{x},\alpha)\\ A_{\mathsf{PCP}}=\{\ \tilde{\mathbf{\Pi}}_*[q]\mid q\in Q\ \}\end{array}\right] > \kappa_{\mathsf{PCP}}\,.$$

By decodability of the index-decodable PCP, it follows that

$$(\mathbf{iD}(\tilde{\pi}_1),\ldots,\mathbf{iD}(\tilde{\pi}_{\mathsf{k}}),\mathbb{x},\mathbf{wD}(\tilde{\Pi}))=(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1),\ldots,\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}),\mathbb{x},\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}'))\in\mathcal{R}\,.$$

Efficiency. We analyze the efficiency parameters of the resulting PCP.

- *Alphabet.* The alphabet used by the system is binary as the error correcting code returns strings of bits.
- *Indexer proof length.* The indexer proof is of length $O(|\mathbb{i}[i]|)$.
- *Prover proof length.* The prover wraps an error-correcting code with constant rate around its proofs. Therefore the proof length is preserved up to constant factors and is $\mathrm{poly}(T)$.
- *Query complexity.* Queries to the indexer proofs remain unchanged. For each of the $O(1)$ queries made to the prover proof of the original PCP, the verifier queries $O(\mathsf{k})$ bits. Hence it makes $O(\mathsf{k})$ queries to the prover proof.
- *Randomness complexity.* The verifier uses the same number of random bits as the original verifier, $O(\log T)$.
- *Indexer running time.* The indexer simply runs $\mathbf{I}_{\mathsf{PCP}}$ and so has running time $\tilde{O}(|\mathbb{i}[i]|)$.
- *Prover running time.* The prover runs $\mathbf{P}_{\mathsf{PCP}}$ and encodes every k-bit symbol of the proof in time that is quasi-polynomial in k. Hence it runs in time $\mathrm{poly}(T)$.
- *Verifier running time.* The verifier runs the original verifier in time $\mathrm{poly}(\mathbb{x},\mathsf{k},\log T)$, and the efficient decoding procedure of the error correcting code to decode $O(1)$ symbols of length k bits. This takes time $\tilde{O}(\mathsf{k})$. Thus, the verifier runs in time $\mathrm{poly}(\mathbb{x},\mathsf{k},\log T)$.
- *Decoder running time.* The index decoder runs the original index decoder and so runs in time $\tilde{O}(|\mathbb{i}[i]|)$. The witness decoder first applies the ECC decoder, and then uses the original witness decoder. It therefore runs in time $\mathrm{poly}(T)$.
- *Adaptivity.* The original index-decodable is non-adaptive, and all that this transformation does is to error-correct queries to the prover proof. Thus the queries are still independent of the proof, and the resulting PCP is non-adaptive.

□

3.9 From round-query IOPs to binary IOPs

We show how to transform a round-query IOP with round-query complexity $\mathsf{q}_{\mathsf{rnd}}$ into a binary IOP with query complexity $O(\mathsf{q}_{\mathsf{rnd}})$.

Theorem 3.9.1. *Let* IOP *be a non-adaptive public-coin round-query IOP for a relation* $\mathcal{R} = \{(\mathbb{x}, \mathbb{w})\}$*. Then there exists a non-adaptive public-coin IOP for* $\mathcal{R}$ *with the parameters below.*

Round-query IOP for $\mathcal{R}$	
Rounds	$\mathsf{k}_{\mathsf{IOP}}$
Proof length	$\mathsf{l}_{\mathsf{IOP}}$
Round queries	$\mathsf{q}_{\mathsf{rnd}}$
Interaction randomness	$\mathsf{r}_{\mathsf{IOP,int}}$
Decision randomness	$\mathsf{r}_{\mathsf{IOP,dc}}$
Soundness error	$O(1)$
Verifier running time	$\mathsf{vt}_{\mathsf{IOP}}$

$\longrightarrow$

IOP for $\mathcal{R}$	
Rounds	$O(\mathsf{k}_{\mathsf{IOP}})$
Alphabet size	2
Proof length	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{l}_{\mathsf{IOP}})$
Queries	$O(\mathsf{q}_{\mathsf{rnd}})$
Interaction randomness	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{r}_{\mathsf{IOP,int}})$
Decision randomness	$O(\log \lvert\mathbb{x}\rvert + \mathsf{r}_{\mathsf{IOP,dc}})$
Soundness error	$O(1)$
Verifier running time	$\mathrm{poly}(\mathsf{vt}_{\mathsf{IOP}})$

This transformation is obtained via two transformations:

- *Local access to interaction randomness.* The IP is transformed into an IOP in which the verifier reads only $O(1)$ bits from each of its random messages but still reads the prover's messages in their entirety. This is described in Section 3.9.1.
- *Local access to prover messages.* The IOP where the verifier has local access to its interaction randomness is transformed into an IOP in which the verifier queries only a few bits of the entire transcript. In our case, we begin with an IOP in which the verifier has local access to its interaction randomness and only reads a small number of rounds overall. We discuss this transformation in Theorem 3.9.7.

Putting together the two transformations yields Theorem 3.9.1.

3.9.1 Local access to interaction randomness

This transformation maps an IP into an IOP where the verifier reads the prover's messages exactly in their entirety but reads only a few bits of each one of its own random messages. In more detail, we prove that a $\mathsf{k}_{\mathsf{IOP}}$-round round-query public-coin IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with round-query complexity $\mathsf{q}_{\mathsf{rnd}}$ and constant soundness error can be transformed into a $O(\mathsf{k}_{\mathsf{IOP}})$-round round-query public-coin IOP $(\mathbf{P}'_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}})$ with round-query complexity $O(\mathsf{q}_{\mathsf{rnd}})$ and constant soundness error, in which the verifier reads $O(\mathsf{q}_{\mathsf{rnd}})$ *bits* from its interaction randomness, but still reads $O(\mathsf{q}_{\mathsf{rnd}})$ messages sent by the prover in their entirety.

Construction 3.9.2. On input $\mathbb{x}$, with parameters $n_z, n_s \in \mathbb{N}$ and (constant) $\gamma \in (0,1)$ the protocol $(\mathbf{P}'_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}})$ works as follows, given an extractor $\mathsf{Ext}\colon \{0,1\}^{n_z} \times \{0,1\}^{n_s} \to \{0,1\}^{\mathsf{r}_{\mathsf{IOP}}}$ with error $\varepsilon_{\mathsf{Ext}}$. The parameters n_z, n_s, $\varepsilon_{\mathsf{Ext}}$ and γ will be fixed during the analysis.

1. The IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ has constant soundness error. Using Theorem 3.3.1, augment $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ such that it has $(1/(|\mathbb{x}| + \mathsf{k}_{\mathsf{IOP}}^2), \gamma/4)$-round-by-round soundness error, and interaction randomness complexity $\mathsf{r}'_{\mathsf{IOP}} = \mathrm{poly}(|\mathbb{x}| + \mathsf{k}_{\mathsf{IOP}})$ and decision randomness complexity $\mathsf{r}'_{\mathsf{IOP},\mathsf{dc}} = O(\mathsf{r}_{\mathsf{IOP},\mathsf{dc}})$, where $\mathsf{r}_{\mathsf{IOP},\mathsf{dc}}$ is the decision randomness of $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$. Since γ is a constant, the augmented IOP has round-query complexity $O(\mathsf{q}_{\mathsf{rnd}})$.

2. For $j = 1, \ldots, \mathsf{k}_{\mathsf{IOP}}$:

   (a) $\mathbf{V}'_{\mathsf{IOP}}$: Send to the prover a random string $z_j \leftarrow \{0,1\}^{n_z}$.

   (b) $\mathbf{P}'_{\mathsf{IOP}}(\mathbb{x}, \mathbb{w}, z_1, s_1, \ldots, z_j)$: Respond with $z'_j \in \{0,1\}^{n_z}$ where (honestly) $z'_j := z_j$.

   (c) $\mathbf{V}'_{\mathsf{IOP}}$: Send to the prover a random seed $s_j \leftarrow \{0,1\}^{n_s}$.

   (d) $\mathbf{P}'_{\mathsf{IP}}(\mathbb{x}, \mathbb{w}, z_1, s_1, \ldots, z_j, s_j)$:

      i. Compute $\alpha_j := \mathsf{Ext}(z'_j, s_j)$.

      ii. Compute $a_j \leftarrow \mathbf{P}_{\mathsf{IOP}}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$.

      iii. Send (s_j, a_j) to the verifier.

3. $\mathbf{V}'^{\mathsf{tr}}_{\mathsf{IOP}}(\mathbb{x})$ where $\mathsf{tr} = \left(z_1, z'_1, s_1, s'_1, a_1, \ldots, z_{\mathsf{k}_{\mathsf{IOP}}}, z'_{\mathsf{k}_{\mathsf{IOP}}}, s_{\mathsf{k}_{\mathsf{IOP}}}, s'_{\mathsf{k}_{\mathsf{IOP}}}, a_{\mathsf{k}_{\mathsf{IOP}}}\right)$. Do the following:

   (a) Sample decision randomness $\alpha_{\mathsf{dc}} \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{IOP},\mathsf{dc}}}$ for $\mathbf{V}_{\mathsf{IOP}}$ and sample random indices $m_z \leftarrow [n_z]$ and $m_s \leftarrow [n_s]$.

   (b) Emulate $\mathbf{V}_{\mathsf{IOP}}^{\alpha_1, a_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IOP}}}, a_{\mathsf{k}_{\mathsf{IOP}}}}(\mathbb{x}; \alpha_{\mathsf{dc}})$ where queries are answered as follows:

      - Any query made by $\mathbf{V}_{\mathsf{IOP}}$ to a prover message a_i is answered by querying the corresponding prover message sent during interaction.
      - If $\mathbf{V}_{\mathsf{IOP}}$ attempts to read α_j then: Check that $z_j[m_z] = z'_j[m_z]$ and $s_j[m_s] = s'_j[m_s]$. If the check fails then reject. Otherwise, pass the string $\alpha_j := \mathsf{Ext}(z'_j, s'_j)$ to $\mathbf{V}_{\mathsf{IOP}}$.

   (c) Accept if and only if $\mathbf{V}_{\mathsf{IOP}}^{\alpha_1, a_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IOP}}}, a_{\mathsf{k}_{\mathsf{IOP}}}}(\mathbb{x}; \alpha_{\mathsf{dc}}) = 1$.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Let tr be a transcript generated in a random execution of the protocol. Since the prover is honest, we have that $z'_j = z_j$ and $s'_j = s_j$, and so the verifier's checks in Item 3b pass with probability 1. Moreover, since the original IP has perfect completeness, and for every j, $a_j := \mathbf{P}_{\mathsf{IOP}}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$, we have that (always) $\mathbf{V}_{\mathsf{IOP}}^{\alpha_1, a_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IOP}}}, a_{\mathsf{k}_{\mathsf{IOP}}}}(\mathbb{x}) = 1$. Therefore, $\mathbf{V}'^{\mathsf{tr}}_{\mathsf{IOP}}(\mathbb{x})$ accepts with probability 1.

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ and a malicious prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. Let E be the event over the verifier's random coins, both interaction and decision, $(z_1, s_1, \ldots, z_{\mathsf{k}_{\mathsf{IOP}}}, s_{\mathsf{k}_{\mathsf{IOP}}}, \alpha_{\mathsf{dc}})$, that there exists some j where the emulated the verifier $\mathbf{V}_{\mathsf{IOP}}$ reads α_j and at least one of the following is true: (i) $\Delta(z'_j, z_j) \geq \gamma$ or; (ii) $\Delta(s'_j, s_j) \geq \gamma$, where $z'_j := \tilde{\mathbf{P}}_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j)$ and $(s'_j, a_j) := \tilde{\mathbf{P}}_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j, s_j)$.

We first show that the probability that the verifier accepts and the event E happens is small:

Claim 3.9.3. *We have that:*

$$\Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \in \mathsf{E}\ \right] \leq 1 - \gamma\,.$$

Proof. For every choice of verifier randomness $(z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \in \mathsf{E}$, there exists some j where the emulated the verifier $\mathbf{V}_{\mathsf{IOP}}$ reads α_j in which either $\Delta(z_j, z'_j) \geq \gamma$ or $\Delta(s_j, s'_j) \geq \gamma$. As a result, one of the checks made by $\mathbf{V}'_{\mathsf{IOP}}$ in Item 3b, causes the verifier to reject with probability at least γ. We conclude the claim by noting that

$$\begin{aligned}&\Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \in \mathsf{E}\ \right]\\ &\quad\leq \Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \mid (z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \in \mathsf{E}\ \right] \leq 1 - \gamma\,.\end{aligned}$$

□

We introduce a new malicious prover $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ for the IOP system that "corrects" any z'_j and s'_j messages sent by the prover that are γ-far from what they are claimed to be. $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ has the following next-message function:

- $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j)$: Compute $z''_j := \tilde{\mathbf{P}}_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j)$. If $\Delta(z''_j, z_j) < \gamma$ then output z''_j and otherwise output z_j.
- $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j, s_j)$: Compute $(s''_j, a_j) := \tilde{\mathbf{P}}_{\mathsf{IOP}}(z_1, s_1, \ldots, z_{j-1}, s_{j-1}, z_j, s_j)$. If $\Delta(s''_j, s_j) < \gamma$ then output (s''_j, a_j) and otherwise output (s_j, a_j).

Notice that $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ and $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ send exactly the same messages in rounds where $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ outputs z'_j and s'_j with $\Delta(z'_j, z_j) < \gamma$ and $\Delta(s'_j, s_j) < \gamma$. Therefore we can reinterpret the event E relative to $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ as follows: $(z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \notin \mathsf{E}$ if for every j where $\mathbf{V}_{\mathsf{IOP}}$ reads α_j and $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ has not "corrected" z'_j or s'_j (by sending z_j or s_j respectively). Whenever the verifier does not read a corrected round, it acts identically to an interaction with $\tilde{\mathbf{P}}_{\mathsf{IP}}$ where it does not read a round that was far from the messages sent by the verifier. Therefore, using this interpretation we have:

$$\begin{aligned}&\Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \notin \mathsf{E}\ \right]\\ &\quad= \Pr\left[\ \langle \tilde{\mathbf{P}}^*_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k_{IOP}}}, s_{\mathsf{k_{IOP}}}, \alpha_{\mathsf{dc}}) \notin \mathsf{E}\ \right]\\ &\quad\leq \Pr\left[\ \langle \tilde{\mathbf{P}}^*_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1\ \right]\end{aligned}$$

We now show that the probability of $\mathbf{V}'_{\mathsf{IOP}}$ accepting when interacting with $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ is small. We begin by showing that $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$-s messages z_j have high min-entropy.

Claim 3.9.4. *For every j: $H_{\mathsf{min}}(Z'_j \mid \neg\mathsf{E}) \geq 0.5n_z$, where Z'_j is the random variable describing the output z'_j of $\tilde{\mathbf{P}}^*_{\mathsf{IP}}$ in a random execution of $\langle \tilde{\mathbf{P}}^*_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle$.*

Proof. Fix a round number j and some string z_j^*. We have that

$$\begin{aligned}
\Pr\left[\ Z'_j = z_j^*\ \right] &\leq \Pr_{z_j}\left[\ \Delta(z_j^*, z_j) < \gamma\ \right] && (3.6)\\
&= |\{\ x' \in \{0,1\}^{n_z} : \Delta(x, x') \leq \gamma\ \}|/2^{n_z}\\
&\leq 2^{-n_z + n_z H(\gamma)} && (3.7)\\
&< 2^{-0.5 n_z}\,. && (3.8)
\end{aligned}$$

Above, Equation 3.6 is due to the fact the output z'_j of $\tilde{\mathbf{P}}^*_{\mathsf{IP}}$ always has Hamming distance at less than γ from z_j. Equation 3.7 true due to Theorem 2.3.5 and Equation 3.8 is true for a small enough constant γ. Then, we get that

$$H_{\mathsf{min}}(Z'_j) \triangleq \min_{z_j^*} -\log \Pr[\ Z'_j = z_j^*\ ] > 0.5 n_z\,.$$

□

Claim 3.9.5. *Suppose that, after amplification in Item 1, the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ has $(\beta_{\mathsf{rbr}}, \delta_{\mathsf{dc}})$-round-by-round soundness with state function* state*. Then for every transcript* tr *generated by an interaction between $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ and $\mathbf{V}'_{\mathsf{IOP}}$ where the verifier is about to make its j-th move such that* $\mathsf{state}(\mathbb{x}, \mathsf{tr}) = 0$*:*

$$\Pr[\ \mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha_j) = 1\ ] \leq (\beta_{\mathsf{rbr}} + \varepsilon_{\mathsf{Ext}}) \cdot 2^{n_s \cdot H(\gamma)}\,,$$

where α_j is drawn as in the protocol description.

Proof. Fix some j and a transcript as in the claim statement. By Theorem 3.9.4, we have that z'_j has min-entropy at least $0.5 n_z$. Thus, by definition of the extractor,

$$|\Pr[\ \mathsf{state}(\mathbb{x}, \mathsf{tr}||\mathsf{Ext}(z'_j, U_{n_s})) = 1\ ] - \Pr[\ \mathsf{state}(\mathbb{x}, \mathsf{tr}||U_{\mathsf{r}'_{\mathsf{IOP}}}) = 1\ ]| \leq \varepsilon_{\mathsf{Ext}}\,,$$

where U_{n_s} and $U_{\mathsf{r}'_{\mathsf{IOP}}}$ are the uniform distributions over bit strings of length n_s and $\mathsf{r}'_{\mathsf{IOP}}$ respectively. Furthermore, by $(\beta_{\mathsf{rbr}}, \delta_{\mathsf{dc}})$-round-by-round soundness of $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$, we have that

$$\Pr[\ \mathsf{state}(\mathbb{x}, \mathsf{tr}||U_{\mathsf{r}'_{\mathsf{IOP}}}) = 1\ ] < \beta_{\mathsf{rbr}}\,.$$

Therefore, the fraction of seeds that cause the state function to change from 0 to 1 is at most $\varepsilon_{\mathsf{Ext}} + \beta_{\mathsf{IOP,rbr}}$. Recall that in the protocol, $\alpha_j := \mathsf{Ext}(z'_j, s'_j)$, i.e., the seed of the extractor is s'_j rather than a uniformly random seed. Recall that $\tilde{\mathbf{P}}^*_{\mathsf{IOP}}$ only outputs messages s'_j with $\Delta(s'_j, s_j) < \gamma$. We say that a seed s_j is *bad* if there exists some s'_j with $\Delta(s'_j, s_j) < \gamma$ such that $\mathsf{state}(\mathbb{x}, \mathsf{tr}||\mathsf{Ext}(z'_j, s'_j)) = 1$. Every point s'_j that inhibits changing of the state function has a ball of size $2^{n_s \cdot H(\gamma)}$ of random seeds that have distance at most γ from it (see Theorem 2.3.5). The total probability of landing on a bad seed is at most the probability that a random seed s_j falls within one of these balls. Therefore the probability that s_j bad is at most $(\varepsilon_{\mathsf{Ext}} + \beta_{\mathsf{IOP,rbr}}) \cdot 2^{n_s \cdot H(\gamma)}$. □

Let β_{rbr} be the round-by-round interaction error of the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ following the augmentation in Item 1. Then $\log 1/\beta_{\text{rbr}} = O(\log(|\mathbb{x}| + \mathsf{k}_{\mathsf{IOP}}^2)) > O(\log(|\mathbb{x} + \mathsf{k}_{\mathsf{IOP}}|)) = \log \mathsf{r}'_{\mathsf{IOP}}$. Therefore, setting $n_z = 4\mathsf{r}'_{\mathsf{IOP}}$, by Theorem 2.3.4, there exists an extractor with error $\varepsilon_{\text{Ext}} = \beta_{\text{rbr}}$, on a source with min entropy $0.5n_z = 2\mathsf{r}'_{\mathsf{IOP}}$ which extracts $\mathsf{r}'_{\mathsf{IOP}}$ bits of randomness. The seed length is $n_s = O(\log 1/\varepsilon_{\text{Ext}}) = O(\log(1/\beta_{\text{rbr}}))$. We therefore have that:

$$\begin{aligned}
\Pr[\langle \tilde{\mathbf{P}}^*_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IP}}(\mathbb{x})\rangle = 1] &\leq \delta_{\mathsf{dc}} + \Pr[\ \exists j : \mathsf{state}(\mathbb{x}, \mathsf{tr}||\alpha_j) = 1\ ] && (3.9)\\
&\leq \delta_{\mathsf{dc}} + \mathsf{k}_{\mathsf{IOP}} \cdot (\beta_{\text{rbr}} + \varepsilon_{\text{Ext}}) \cdot 2^{n_s \cdot H(\gamma)} && (3.10)\\
&\leq \delta_{\mathsf{dc}} + \mathsf{k}_{\mathsf{IOP}} \cdot 2\beta_{\text{rbr}} \cdot 2^{O(\log(1/\beta_{\text{rbr}})) \cdot H(\gamma)} && (3.11)\\
&\leq \delta_{\mathsf{dc}} + \mathsf{k}_{\mathsf{IOP}} \cdot \sqrt{\beta_{\text{rbr}}} && (3.12)\\
&\leq \gamma/4 + \mathsf{k}_{\mathsf{IOP}}/\sqrt{|\mathbb{x}| + \mathsf{k}_{\mathsf{IOP}}^2} < \gamma/2\,. && (3.13)
\end{aligned}$$

Equation 3.9 follows from the fact that the verifier $\mathbf{V}'_{\mathsf{IP}}$ accepts only if $\mathbf{V}_{\mathsf{IP}}$ accepts given $\mathbb{x}$, prover messages $a_1, \ldots, a_{\mathsf{k}_{\mathsf{IOP}}}$ and verifier randomness $\alpha_1, \ldots . \alpha_{\mathsf{k}_{\mathsf{IOP}}}$. By the round-by-round soundness of the original IP, since $\mathsf{state}(\mathbb{x}, \emptyset) = 0$ (which follows from the fact that $\mathbb{x} \notin L$), in order for the verifier to accept, it must be that the value of the state function changed from 0 to 1 in some round. Equation 3.10 is true by applying the union bound and Theorem 3.9.5. We have Equation 3.11 by noting that we set $\varepsilon_{\text{Ext}} = \beta_{\text{rbr}}$ and $n_s = O(\log(1/\beta_{\text{rbr}}))$. Equation 3.12 holds for a small enough constant $\gamma > 0$, and, finally, Equation 3.13 holds by the definitions of δ_{dc} and β_{rbr} and for large enough values of $|\mathbb{x}|$.

Thus we have that

$$\Pr\left[\begin{array}{c}\langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \\ \wedge\ (z_1, s_1, \ldots, z_{\mathsf{k}_{\mathsf{IOP}}}, s_{\mathsf{k}_{\mathsf{IOP}}}, \alpha_{\mathsf{dc}}) \notin \mathsf{E}\end{array}\right] \leq \Pr\left[\ \langle \tilde{\mathbf{P}}^*_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1\ \right] \leq \gamma/2\,.$$

Putting this fact together with Theorem 3.9.3, we have that

$$\begin{aligned}
\Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1\ \right] &= \Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k}_{\mathsf{IOP}}}, s_{\mathsf{k}_{\mathsf{IOP}}}, \alpha_{\mathsf{dc}}) \in \mathsf{E}\ \right]\\
&\quad + \Pr\left[\ \langle \tilde{\mathbf{P}}_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}}(\mathbb{x})\rangle = 1 \wedge (z_1, s_1, \ldots, z_{\mathsf{k}_{\mathsf{IOP}}}, s_{\mathsf{k}_{\mathsf{IOP}}}, \alpha_{\mathsf{dc}}) \notin \mathsf{E}\ \right]\\
&\leq 1 - \gamma + \gamma/2 = 1 - \gamma/2\,.
\end{aligned}$$

Thus the verifier accepts with constant probability.

Complexity measures. We analyze the efficiency parameters of the resulting round-query IOP:

- *Rounds.* The round-query IOP has $2\mathsf{k}_{\mathsf{IOP}}$ rounds.
- *Proof length.* We first amplify the protocol, giving polynomial overhead to all messages. In addition to the original prover messages, the prover also sends z'_j and s'_j. Therefore the proof length is $\text{poly}(|\mathbb{x}|, \mathsf{l}_{\mathsf{IOP}})$.

- *Round queries.* $\mathbf{V}'_{\mathsf{IOP}}$ queries $O(\mathsf{q}_{\mathsf{rnd}})$ rounds.
- *Query complexity to randomness.* The verifier queries s_j and z_j in $O(1)$ locations.
- *Randomness complexity.* $\mathbf{V}'_{\mathsf{IP}}$ generates $n_z + n_s = \mathrm{poly}(\mathsf{r}_{\mathsf{IP}}, |\mathbb{x}|)$ bits in every round.
- *Decision randomness.* $\mathbf{V}'_{\mathsf{IP}}$ chooses decision randomness $\mathsf{r}'_{\mathsf{IOP},\mathsf{dc}} = O(\mathsf{r}_{\mathsf{IOP},\mathsf{dc}})$ and then uses $\log n_z + \log n_s = O(\log |\mathbb{x}| + \log \mathsf{k}_{\mathsf{IOP}})$ bits of decision randomness. All-in-all $O(\mathsf{r}_{\mathsf{IOP},\mathsf{dc}} + \log |\mathbb{x}|)$.
- *Verifier running time.* $\mathbf{V}'_{\mathsf{IP}}$ runs the original IOP verifier for polynomially many repetitions, generates a few random strings and runs the extractor. Its running time is therefore polynomially related to the running time of $\mathbf{V}_{\mathsf{IP}}$.
- *Adaptivity.* $\mathbf{V}'_{\mathsf{IP}}$ makes non-adaptive queries to its interaction randomness. Therefore, if $\mathbf{V}_{\mathsf{IOP}}$ was non-adaptive, then so is $\mathbf{V}'_{\mathsf{IOP}}$.

3.9.2 Local access to prover messages

In this transformation, we take a round-query IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with round-query complexity $\mathsf{q}_{\mathsf{rnd}}$ where the verifier reads $O(1)$ bits from the interaction randomness of the rounds that it queries but reads the entire prover messages, and transform it into a binary IOP $(\mathbf{P}'_{\mathsf{IOP}}, \mathbf{V}'_{\mathsf{IOP}})$ with query complexity $O(\mathsf{q}_{\mathsf{rnd}})$. We show the transformation for IPs (following [ACY22a]), and explain later how this translates to round-query IOPs.

Definition 3.9.6. *Given a k_{IP}-round public-coin IP $\mathsf{IP} = (\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$, define the multi-indexed relation*

$$\Psi(\mathbf{V}_{\mathsf{IP}}) := \{(a_1, \ldots, a_{\mathsf{k}_{\mathsf{IP}}}, (\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IP}}}), \bot) \;\mid\; \mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, a_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IP}}}, a_{\mathsf{k}_{\mathsf{IP}}}) = 1\} \,.$$

Here $\mathbb{x}$ corresponds to the common input instance to the IP prover and IP verifier, $\alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{IP}}}$ correspond to verifier (random) messages, and $a_1, \ldots, a_{\mathsf{k}_{\mathsf{IP}}}$ correspond to prover messages.

Theorem 3.9.7. *Suppose that:*
- $\mathsf{IP} = (\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ *is a public-coin IP for a relation $\mathcal{R}$; and*
- $\mathsf{PCP} = (\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ *is an index-decodable PCP for the multi-indexed relation $\Psi(\mathbf{V}_{\mathsf{IP}})$.*

Then Theorem 3.9.9 is a $(\mathsf{k}_{\mathsf{IP}} + 1)$-round public-coin IOP for $\mathcal{R}$ with the parameters below.

IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ for $\mathcal{R}$	
Rounds	k_{IP}
Prover-to-verifier communication	l_{IP}
Interaction randomness	$\mathsf{r}_{\mathsf{IP},\mathsf{int}}$
Decision randomness	$\mathsf{r}_{\mathsf{IP},\mathsf{dc}}$
Soundness error	β_{IP}
Verifier running time	$\mathsf{vt}_{\mathsf{IP}}$

+

Index-Decodable PCP for $\Psi(\mathbf{V}_{\mathsf{IP}})$	
Indexer proof length	$\mathsf{l}_{\mathsf{PCP},\mathbf{I}}$
Proof length	$\mathsf{l}_{\mathsf{PCP},\mathbf{P}}$
Queries per proof	$\mathsf{q}_{\mathsf{PCP}}$
Randomness	$\mathsf{r}_{\mathsf{PCP}}$
Decodability bound	κ_{PCP}
Verifier running time	$\mathsf{vt}_{\mathsf{PCP}}$

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Rounds	k_{IP}
Proof length	$k_{IP} \cdot l_{PCP,I} + l_{PCP,P} \cdot 2^{r_{IP,dc}}$
Queries per round	q_{PCP}
Total round randomness	$r_{IP,int}$
Decision randomness	$r_{PCP} + r_{IP,dc}$
Soundness error	$\beta_{IP} + \kappa_{PCP}$
Verifier running time	vt_{PCP}

Moreover, if $\mathbf{iD}_{\mathsf{PCP}}$ *is efficient then the transformation maintains computational soundness (if* IP *has computational soundness error* β_{IP}*, then* IOP *has computational soundness error* $\beta_{IP} + \kappa_{PCP}$*).*

Remark 3.9.8. The transformation in Theorem 3.9.1 can be modified to preserve the verifier's randomness query complexity if the verifier is non-adaptive with respect to the queries it makes to its interaction randomness. Suppose that the verifier reads q bits from its own messages. Then we define a multi-indexed relation that consists of tuples:

$$\left(\mathbb{i}[1], \ldots, \mathbb{i}[k], \mathbb{x}, \mathbb{w}\right) = \left(a_1, \ldots, a_k, (\mathbb{x}, b_1, \ldots, b_q, \alpha_{dc}), \bot\right)$$

such that given decision randomness α_{dc} the IP verifier $\mathbf{V}_{IP}$ accepts given instance $\mathbb{x}$, decision randomness α_{dc}, prover messages $(a_1, \ldots, a_k)$, and $(b_1, \ldots, b_q)$ as answers to queries to its own interaction randomness. Given a multi-indexed PCP for this relation, the IP to IOP transformation is identical to the one in Theorem 3.9.9, except that at the end, after the verifier chooses decision randomness, it also queries its own randomness to get bits $b_1, \ldots, b_q$, and these replace $\alpha_1, \ldots, \alpha_{k_{IP}}$ as explicit inputs to the index-decodable PCP verifier.

We now prove Theorem 3.9.7; we describe the construction and then analyze it.

Construction 3.9.9. The IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ receives an instance $\mathbb{x}$ and the (honest) IOP prover $\mathbf{P}_{\mathsf{IOP}}$ receives $\mathbb{x}$ and a witness $\mathbb{w}$. They interact as follows.

1. For every round $i \in [k_{IP}]$:
   (a) $\mathbf{V}_{\mathsf{IOP}}$ sends a uniformly random string α_i as sampled by $\mathbf{V}_{IP}$;
   (b) $\mathbf{P}_{\mathsf{IOP}}$ computes $a_i \leftarrow \mathbf{P}_{IP}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_i)$ and sends $\pi_i := \mathbf{I}_{PCP}(a_i)$.
2. $\mathbf{P}_{\mathsf{IOP}}$ sends, for every $\alpha_{dc} \in \{0,1\}^{r_{IP,dc}}$, $\Pi_{\alpha_{dc}} := \mathbf{P}_{PCP}(a_1, \ldots, a_{k_{IP}}, (\mathbb{x}, \alpha_1, \ldots, \alpha_{k_{IP}}, \alpha_{dc}), \bot)$.
3. $\mathbf{V}_{\mathsf{IOP}}$ (given oracle access to $\tilde{\pi}_1, \ldots, \tilde{\pi}_{k_{IP}}$ and $\tilde{\Pi} = \{\tilde{\Pi}_{\alpha_{dc}}\}_{\alpha_{dc}}$) samples decision randomness α_{dc} and PCP randomness α_{PCP} and checks that

$$\mathbf{V}_{PCP}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{k_{IP}}, \tilde{\Pi}_{\alpha_{dc}}}((\mathbb{x}, \alpha_1, \ldots, \alpha_{k_{IP}}, \alpha_{dc}); \alpha_{PCP}) = 1\,.$$

Proof of Theorem 3.9.1. First we argue completeness, then argue soundness, and finally analyze efficiency measures.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. The strings $a_1, \ldots, a_{\mathsf{k_{IP}}}$ are computed by running the honest IP prover $\mathbf{P}_{\mathsf{IP}}$ given $(\mathbb{x}, \mathbb{w})$. By the (perfect) completeness of the IP,

$$\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, a_1, \ldots, \alpha_{\mathsf{k_{IP}}}, a_{\mathsf{k_{IP}}}; \alpha_{\mathsf{dc}}) = 1\,,$$

with probability 1 over $\mathbf{V}_{\mathsf{IP}}$'s randomness $\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}}$. Hence,

$$(a_1, \ldots, a_{\mathsf{k_{IP}}}, (\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}}), \bot) \in \Psi(\mathbf{V}_{\mathsf{IP}})\,,$$

with probability 1 over $\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}}$. Moreover, by the (perfect) completeness of the index-decodable PCP, $\mathbf{V}_{\mathsf{PCP}}$ accepts with probability 1 (over α_{PCP}) when given access to the indexer proofs π_i obtained from the indexes a_i via $\mathbf{I}_{\mathsf{PCP}}$ and the prover proof $\Pi_{\alpha_{\mathsf{dc}}}$ output by the PCP prover $\mathbf{P}_{\mathsf{PCP}}$. We conclude that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability 1, as desired.

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ ($L(\mathcal{R})$ is the language implied by the relation $\mathcal{R}$) and let $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ be a malicious IOP prover. In the following, we let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}})$ denote a list of $\mathsf{k_{IP}}$ verifier messages and $\boldsymbol{\alpha}_i = (\alpha_1, \ldots, \alpha_i)$ a prefix of length i. Let E be the event over the verifier's coins $\boldsymbol{\alpha}$ that

$$(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k_{IP}}}), (\mathbb{x}, \boldsymbol{\alpha}), \bot) \in \Psi(\mathbf{V}_{\mathsf{IP}})\,,$$

where $\tilde{\pi}_i := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_i)$ for any $i \in \{1, \ldots, \mathsf{k_{IP}} - 1\}$ and $(\tilde{\pi}_{\mathsf{k_{IP}}}, \tilde{\Pi}) := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}})$ where $\tilde{\Pi} = \{\tilde{\Pi}_{\alpha_{\mathsf{dc}}}\}_{\alpha_{\mathsf{dc}} \in [\mathsf{r_{IP,dc}}]}$. By the definition of $\Psi(\mathbf{V}_{\mathsf{IP}})$, $(\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}}) \in E$ if and only if the proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k_{IP}}}$ can be decoded into messages that make the IP verifier accept:

$$\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \alpha_1, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \alpha_{\mathsf{k_{IP}}}, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k_{IP}}}); \alpha_{\mathsf{dc}}) = 1\,.$$

Using the following claims, by the law of total probability we conclude that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\kappa_{\mathsf{PCP}} + \beta_{\mathsf{IP}}$ as desired.

Claim 3.9.10. *We have that:*

$$\Pr_{\boldsymbol{\alpha}, \alpha_{\mathsf{PCP}}}\left[\mathbf{V}_{\mathsf{IOP}}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k_{IP}}}, \tilde{\Pi}}(\mathbb{x}, \boldsymbol{\alpha}; \alpha_{\mathsf{PCP}}) = 1 \ \wedge\ E \,\middle|\, \begin{array}{r} \tilde{\pi}_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_1) \\ \vdots \\ \tilde{\pi}_{\mathsf{k_{IP}}-1} \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}-1}) \\ (\tilde{\pi}_{\mathsf{k_{IP}}}, \tilde{\Pi}) \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}}) \end{array}\right] \leq \beta_{\mathsf{IP}}\,.$$

Proof. Consider the malicious IP prover $\tilde{\mathbf{P}}_{\mathsf{IP}}$ that simulates $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ by passing it the verifier's messages, decoding the proof that $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ sends in return, and sending the decoded message to the IP verifier. More formally, in round $1 \leq i \leq \mathsf{k_{IP}}$, letting $\alpha_1, \ldots, \alpha_i$ be the verifier messages up to this point, $\tilde{\mathbf{P}}_{\mathsf{IP}}$ computes $\pi_i := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1, \ldots, \alpha_i)$ and sends $a_i := \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_i)$ as its message to the IP verifier (in round $\mathsf{k_{IP}}$, $\tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}})$ outputs in addition to $\tilde{\pi}_{\mathsf{k_{IP}}}$ also a proof $\tilde{\Pi}$, but this proof is ignored by $\tilde{\mathbf{P}}_{\mathsf{IP}}$).

Let $\varepsilon_{\mathsf{IP}}$ be the probability that the IP verifier $\mathbf{V}_{\mathsf{IP}}$ accepts when interacting with $\tilde{\mathbf{P}}_{\mathsf{IP}}$:

$$\varepsilon_{\mathsf{IP}} = \Pr_{\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}, \alpha_{\mathsf{dc}}}\left[\mathbf{V}_{\mathsf{IP}}(\alpha_1, a_1, \ldots, \alpha_{\mathsf{k_{IP}}}, a_{\mathsf{k_{IP}}}; \alpha_{\mathsf{dc}}) = 1 \,\middle|\, \begin{array}{r} a_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IP}}(\alpha_1) \\ \vdots \\ a_{\mathsf{k_{IP}}} \leftarrow \tilde{\mathbf{P}}_{\mathsf{IP}}(\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}}) \end{array}\right].$$

By definition, whenever $\boldsymbol{\alpha} \in E$ the IP verifier accepts given messages $(a_1, \ldots, a_{\mathsf{k_{IP}}})$ decoded from the proofs that the prover sent given instance $\mathbb{x}$ and verifier messages $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{\mathsf{k_{IP}}})$ (i.e., $a_i := \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_i)$ where $\tilde{\pi}_i := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\alpha_1, \ldots, \alpha_i)$). This is precisely how the malicious IP prover computes its own messages. Hence, for every instance $\mathbb{x}$ and verifier messages $\boldsymbol{\alpha}$ for which simultaneously the IOP verifier accepts and also $\boldsymbol{\alpha} \in E$, the malicious IP prover $\tilde{\mathbf{P}}_{\mathsf{IP}}$ makes the IP verifier accept. By soundness of the IP, the verifier accepts $\mathbb{x} \notin L(\mathcal{R})$ with probability at most β_{IP} over its random messages regardless of what the malicious IP prover does. Thus we conclude that:

$$\Pr_{\boldsymbol{\alpha},\alpha_{\mathsf{PCP}}}\left[\begin{array}{l|r} \mathbf{V}_{\mathsf{IOP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k_{IP}}},\tilde{\Pi}}(\mathbb{x},\boldsymbol{\alpha};\alpha_{\mathsf{PCP}}) = 1 \ \wedge \ E & \tilde{\pi}_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_1) \\ & \vdots \\ & \tilde{\pi}_{\mathsf{k_{IP}}-1} \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}-1}) \\ & (\tilde{\pi}_{\mathsf{k_{IP}}},\tilde{\Pi}) \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}}) \end{array}\right] \leq \varepsilon_{\mathsf{IP}} < \beta_{\mathsf{IP}}\,.$$

□

Claim 3.9.11. *We have that:*

$$\Pr_{\boldsymbol{\alpha},\alpha_{\mathsf{PCP}}}\left[\begin{array}{l|r} \mathbf{V}_{\mathsf{IOP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k_{IP}}},\tilde{\Pi}}(\mathbb{x},\boldsymbol{\alpha};\alpha_{\mathsf{PCP}}) = 1 \ \wedge \ \neg E & \tilde{\pi}_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_1) \\ & \vdots \\ & \tilde{\pi}_{\mathsf{k_{IP}}-1} \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}-1}) \\ & (\tilde{\pi}_{\mathsf{k_{IP}}},\tilde{\Pi}) \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}}) \end{array}\right] \leq \kappa_{\mathsf{PCP}}(|\mathbb{x}| + \mathsf{r}_{\mathsf{IP,int}} + \mathsf{r}_{\mathsf{IP,dc}})\,.$$

Proof. Assume towards contradiction that the claim does not hold. There must exist $\boldsymbol{\alpha} \notin E$ such that

$$\Pr_{\alpha_{\mathsf{PCP}}}\left[\begin{array}{l|r} \mathbf{V}_{\mathsf{IOP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_k,\tilde{\Pi}}(\mathbb{x},\boldsymbol{\alpha};\alpha_{\mathsf{PCP}}) = 1 & \tilde{\pi}_1 \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_1) \\ & \vdots \\ & \tilde{\pi}_{\mathsf{k_{IP}}-1} \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}-1}) \\ & (\tilde{\pi}_{\mathsf{k_{IP}}},\tilde{\Pi}) \leftarrow \tilde{\mathbf{P}}_{\mathsf{IOP}}(\boldsymbol{\alpha}_{\mathsf{k_{IP}}}) \end{array}\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}| + \mathsf{r}_{\mathsf{IP,int}} + \mathsf{r}_{\mathsf{IP,dc}})\,.$$

The IOP verifier accepts if and only if the underlying PCP verifier accepts. This means that the PCP verifier accepts with probability greater than $\kappa_{\mathsf{PCP}}(|\mathbb{x}| + \mathsf{r}_{\mathsf{IP,int}} + \mathsf{r}_{\mathsf{IP,dc}})$ (the knowledge bound of the PCP). Thus, by decodability of the index-decodable PCP, we get that:

$$(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k_{IP}}}), (\mathbb{x},\boldsymbol{\alpha}), \perp) \in \Psi(\mathbf{V}_{\mathsf{IP}})\,.$$

This contradicts the assumption that $\boldsymbol{\alpha} \notin E$. □

Notice that the prover $\tilde{\mathbf{P}}_{\mathsf{IP}}$ above simply runs the malicious IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ and the index decoder $\mathbf{iD}_{\mathsf{PCP}}$. If both $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ and $\mathbf{iD}_{\mathsf{PCP}}$ run in polynomial time, then so does $\tilde{\mathbf{P}}_{\mathsf{IP}}$. Hence if this is the case, and the IP is *computationally* sound (sound against efficient adversaries) then so is the resulting IOP.

Complexity measures. The number of rounds is $\mathsf{k_{IP}}$. The IOP verifier uses $\mathsf{r}_{\mathsf{IP,int}}$ during interaction and $\mathsf{r}_{\mathsf{PCP}} + \mathsf{r}_{\mathsf{IP,dc}}$ random bits in the decision phase. The IOP verifier

makes q_{PCP} queries to its oracles when running the PCP verifier. The IOP verifier's running time is vt_{PCP} since it runs the PCP verifier. The IOP prover generates k_{IP} indexer proofs each of length $l_{PCP,\mathbf{I}}$, and $2^{r_{dc}}$ prover proofs of length $l_{PCP,\mathbf{P}}$.

□

3.9.2.1 Extending to round-query IOPs

The transformation involves simulating the round-query IOP, except that $\mathbf{P}'_{IOP}$ encodes the messages of $\mathbf{P}_{IOP}$ using an ID-PCP. In the final round, $\mathbf{P}'_{IOP}$ sends, for every choice of decision randomness of $\mathbf{V}_{IOP}$, a proof convincing that $\mathbf{V}_{IOP}$ would have accepted, having chosen this decision randomness. The new IOP verifier $\mathbf{V}'_{IOP}$ then chooses decision randomness and runs the ID-PCP verifier to verify the proof. Executing the ID-PCP verifier involves reading the relevant bits of the verifier interaction randomness, $O(1)$ bits of the final prover message, and $O(1)$ bits from the encoding of each prover message that $\mathbf{V}_{IOP}$ would have queried using this decision randomness. Hence, $\mathbf{V}'_{IOP}$ has (total) query complexity $O(q_{rnd})$.

Chapter 4

IOPs with inverse polynomial soundness error

4.1 Introduction

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 4.1.1 (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].

4.1.1 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 4. *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 4 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 4.1.2 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 4 applies for general NP languages, for the NP-complete language 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 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 4 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,

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

and, in particular, they often play a role in the design of probabilistic proofs. The code $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ is the set of functions $f\colon \mathcal{L} \to \mathbb{F}$ that are evaluations of polynomials of degree (at most) d. An IOP of proximity (IOPP) for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ enables a prover to convince a verifier with oracle access to f that f is close (in relative Hamming distance) to a codeword in $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ by engaging in a multi-round interaction where the prover sends additional oracles and the verifier sends randomness. We provide an IOPP for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ with small soundness error and small query complexity.

Theorem 5 (informal). *Let $\mathbb{F}$ be a "nice" field. There is a public-coin IOPP for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ 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 $O(|\mathcal{L}|/\rho)$ over $\mathbb{F}$, and query complexity $O(\log\log d)$; here $\rho := \frac{d+1}{|\mathcal{L}|}$ is the code rate.*

A formal statement and proof appear in Section 4.5.

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; BGHSV06a; BGKS20; BCIKS20]).

A building block of independent interest on the way to Theorem 5 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 6 (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 4.7.3.

4.1.2 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

[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.

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 4, 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 $O(\log\log n)$.

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 $O(\log\log d)$ recursive steps, the soundness error is $1 - \frac{1}{\mathrm{polylog}(d)}$ even for functions that are far from the code. To compensate for this, soundness is amplified by repetition, yielding query complexity $\mathrm{polylog}(d)$.

Our proximity test overcomes both of these issues, using only $O(\log\log d)$ recursive steps and query complexity $O(\log\log d)$ while maintaining high soundness throughout. The proof length in our protocol is slightly higher at $O(|\mathcal{L}|/\rho)$, com-

pared to FRI, which has proof length $O(|\mathcal{L}|)$, and [BS08], which has proof length $\widetilde{O}(|\mathcal{L}|)$.

4.2 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 4. See Section 4.6 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

[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).

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 $O(\log\log d)$ 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.
- 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 4.2.1, we outline how we compile a poly-IOP into an IOP using proximity testing for Reed–Solomon codes. In Section 4.2.2, we describe our poly-IOP for Reed–Solomon codes. Finally, in Section 4.2.3, we sketch the proof of Theorem 5, showing how to combine the tools developed in Section 4.2.1 and Section 4.2.2 to construct a proximity test for Reed–Solomon codes.

4.2.1 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 $\mathcal{L}$ 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 $\mathcal{L}$, where $\rho = (d+1)/|\mathcal{L}|$. 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 $\mathcal{R}$ 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 $\mathcal{R}$ 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]

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

- *A proximity test for* $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ *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 $\mathcal{R}$ *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* $O(\mathsf{k} \cdot |\mathcal{L}| + \mathsf{l}_{\mathsf{prx}})$*, and query complexity* $O(\mathsf{q} + \mathsf{k} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\Pi})$*. Here* $\rho := (d+1)/|\mathcal{L}|$ *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 $\hat{f}_i \in \mathbb{F}^{\leq d_i}[X]$ 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 $f_i \colon \mathcal{L} \to \mathbb{F}$, the *evaluation* of $\hat{f}_i$ over the proximity test domain $\mathcal{L}$; that is, $f_i \in \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_i]$ is such that $f_i(\mathcal{L}) = \hat{f}_i(\mathcal{L})$. 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.

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 $\hat{f}_i(\mathcal{L})$, which implies that we need $\mathcal{L} = \mathbb{F}$. 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 $\mathcal{L}$, 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 $\deg(\hat{f}) - |S|$. 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 $\mathcal{L}$ disagree with p, then the quotient function $\mathsf{Quotient}(f, S, p)$ is far from low-degree over $\mathcal{L}$.

Lemma 2. *Suppose that for every polynomial $\hat{f}$ of degree at most d with $\Delta(f, \hat{f}(\mathcal{L})) \leq \delta$ 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 $\mathcal{L}$ 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 $\mathcal{L}$. 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 $\mathcal{L}$. 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, 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 $\mathcal{L}$ 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

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

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 $f_i \colon \mathcal{L} \to \mathbb{F}$, the evaluation of $\hat{f}_i$ over $\mathcal{L}$. 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 $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_i]$, 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 $\mathcal{L}$. 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 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.

4.2.1.1 Summary: compiling poly-IOPs to IOPs

We summarize our compiler from poly-IOP to IOP. Let $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$ 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-

[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.

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 $\mathbf{P}_{\mathsf{poly}}$ to obtain a polynomial $\hat{f}_i \in \mathbb{F}^{\leq d}[X]$, and sends $f_i \colon \mathcal{L} \to \mathbb{F}$ (the evaluation of $\hat{f}_i$ on $\mathcal{L}$).
   (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 $\mathbf{V}_{\mathsf{poly}}$ sends in the i-th round.
2. For every i, the prover computes the set of queries S_i that $\mathbf{V}_{\mathsf{poly}}$ 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 $r_1, \ldots, r_{\mathsf{k}}$.
4. The prover and verifier run the proximity test for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d - \mathsf{q} - 1]$ on the function
$$h(x) = \sum_{i \in [\mathsf{k}]} r_i \cdot g_i(x)\,,$$
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 $\mathbf{V}_{\mathsf{poly}}$ 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.

4.2.1.2 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 $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$: given a function f to be tested, the prover sends a polynomial $\hat{h} \in \mathbb{F}^{\leq d}[X]$. The verifier then picks $a \leftarrow \mathcal{L}$ 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 $\mathcal{L}$. If we have an IOPP for testing RS codes over a different domain $\mathcal{L}'$, then we can apply Lemma 1 with the poly-IOPP for $\mathcal{C}$ using the IOPP for RS codes over $\mathcal{L}'$ to get an IOPP that works over $\mathcal{L}$ as desired. See Section 4.7.2 for more details.

4.2.2 poly-IOPP for RS codes

We sketch a poly-IOPP for the code $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ (evaluations on the domain $\mathcal{L}$ 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 $\sqrt{|\mathbb{F}|} > |G| \geq \text{poly}(d, |\mathcal{L}|)$.

Lemma 3 (informal)**.** *There is a poly-IOPP for* $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ *with perfect completeness, soundness error (roughly)* $2\sqrt{1-\delta} + \text{poly}(d, 1/\rho)/|\mathbb{F}|$ *for* δ*-far functions, round complexity* $O(1)$*, and total query complexity* $O(1)$*; here* $\rho := (d+1)/|\mathcal{L}|$*. The prover sends* $O(|\mathcal{L}|/\rho)$ *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 $\mathcal{L}$ is a multiplicative subgroup of $\mathbb{F}^*$ whose order is divisible by m. Our protocol, however, works for any degree d and evaluation domain $\mathcal{L}$.

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 ([BS08]). *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 $\deg_X(\hat{Q}) = m - 1$ and $\deg_Y(\hat{Q}) \leq m - 1$ 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 $\hat{Q}$: $D := \{(\hat{q}(j), j) \mid j \in \mathcal{L}\}$.

Let $f \colon \mathcal{L} \to \mathbb{F}$ be a function claimed to be in $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ and $Q \colon D \to \mathbb{F}$ be the bivariate function such that, for every $x \in \mathcal{L}$, $Q(\hat{q}(x), x) = f(x)$. 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 $d^* \cdot \deg(\hat{q}) + d^* = m^2 - 1 = d$ agrees with f on a $(1-\delta)$-fraction of the points of $\mathcal{L}$ (indeed, for a $(1-\delta)$-fraction of $\mathcal{L}$ 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.

4.2.2.1 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 $\mathcal{L}_X := \{i \mid \exists\, j \text{ s.t. } (i, j) \in D\}$ and, for every $i \in \mathcal{L}_X$, let $\mathcal{L}_Y^{(i)} := \{j \mid (i, j) \in D\}$ (i.e., $\mathcal{L}_Y^{(i)}$ is the set of all elements $j \in \mathcal{L}$ such that $j^m = i$ over the multiplicative subgroup $\mathcal{L}$). Since $\mathcal{L}$ (as described in the previous section) is a multiplicative subgroup whose order is divisible by m, each set $\mathcal{L}_Y^{(i)}$ has size $|D|/|\mathcal{L}_X|$.

Our protocol begins with the prover sending the rows of the polynomial $\hat{Q}$ corresponding to the domains $\mathcal{L}_Y^{(i)}$; each row is a polynomial of degree d^*. That is, the prover sends polynomials $(\hat{r}_i)_{i \in \mathcal{L}_X}$ 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 $c_j \colon \mathcal{L}_X \to \mathbb{F}$ is defined as $c_j(i) := \hat{r}_i(j)$ for $i \in \mathcal{L}_X$. If the prover is honest, then c_j is the evaluation of $\hat{Q}(\cdot, j) \in \mathbb{F}^{\leq d^*}[X]$ over $\mathcal{L}_X$, and

so $c_j \in \mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, d^*]$. Let $M(i,j) = \hat{r}_i(j) = c_j(i)$ for $i \in \mathcal{L}_X$ and $j \in \mathbb{F}$ be the description of the rows and columns as a matrix on $\mathcal{L}_X \times \mathbb{F}$.

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 $j \in \mathcal{L}_Y^{(i)}$), 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 4.2.2.2 we explain how we test consistency of the rows with a low-degree bivariate polynomial and then in Section 4.2.2.3 we explain how to test consistency with Q in the same area.

4.2.2.2 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 $\mathcal{L}_X$ is equal to c_j (i.e., $\hat{v}_j(a) = c_j(a)$ for every $a \in \mathcal{L}_X$). The verifier samples a random point $a \leftarrow \mathcal{L}_X$ 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 $(\hat{r}_i)_{i \in \mathcal{L}_X}$ 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 at least d^*+1. 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 $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, d^*]$, 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 $W \subseteq \mathcal{L}_X$ such that for every $j \in H$ there exists $u_j \in \mathcal{C}_X$ where $c_j(W) = u_j(W)$. For coefficients $\vec{r} = (r_j)_{j\in H}$ let $c_{\vec{r}} = \sum_{j\in H} r_j \cdot c_j$, and let $\Delta(f,g)$ denote the relative Hamming distance between f and g.

Theorem 7 (Proximity gap for RS codes [BCIKS20]; informal)**.** *If* $\Pr_{\vec{r}}[\Delta(c_{\vec{r}}, \mathcal{C}_X) \le 1-\mu] > \mathsf{err}^* = O(\mathrm{poly}(|\mathcal{L}_X|)/|\mathbb{F}|)$ *then there exists a set* $W \subseteq \mathcal{L}_X$ *of size at least* $\mu \cdot |\mathcal{L}_X|$ *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 $c_{\vec{r}}$ 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* $W \subseteq \mathcal{L}_X$ *on which* $(c_j)_{j\in H}$ *have correlated agreement and* $|W| \ge \mu \cdot |\mathcal{L}_X| \ge d^*$*, 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 $\vec{r}$, and the prover sends a polynomial $\hat{v}$ claimed to be equal to the low-degree extension of the column $c_{\vec{r}}$. The verifier compares the two at a random location in $\mathcal{L}_X$. This requires the verifier to compute $c_{\vec{r}}(i)$ 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 $c_{\vec{r}}(i) = \sum_{j \in H} r_j \cdot c_j(i) = \sum_{j \in H} r_j \cdot \hat{r}_i(j)$. 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 $\sum_{j \in H} r_j \cdot \hat{r}_i(j) = \gamma$, 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).

4.2.2.3 Ensuring simultaneous consistency with Q

So far, we have forced the prover to send rows $(\hat{r}_i)_{i \in \mathcal{L}_X}$ 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 $\mathcal{L}_Y^{(i)}$ (the locations on which $\hat{r}_i$ and Q coincide).

If many rows $(\hat{r}_i)_{i \in \mathcal{L}_X}$ 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 $\mathcal{L}_Y^{(i)}$ has the same number of points, sampling a uniform $i \in \mathcal{L}_X$ and then uniform $j \in \mathcal{L}_Y^{(i)}$ 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 7, 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.

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

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 7 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 $j \in \mathcal{L}_Y^{(i)}$. 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 7, which will give us this extra guarantee, which we call proximity gaps for subsets.

Proximity gaps for subsets. Our strengthening of Theorem 7 essentially shows that if (with probability $\mathsf{err}^\star$) a random linear combination agrees with a codeword on a fixed set S, then there exists W as in Theorem 7, with the additional guarantee that $W \subseteq S$.

Theorem 8 (Theorem 4.3.4; informal). *Fix $S \subseteq \mathcal{L}_X$ with $|S| \geq \mu \cdot |\mathcal{L}_X|$. If*

$$\Pr_{r_1,\ldots,r_{|H|}}\left[\begin{array}{l}\exists\, T \subseteq S \\ |T| \geq \mu \cdot |\mathcal{L}_X|\end{array} \;:\; \exists\, u \in \mathcal{C}_X,\ u(T) = \textstyle\sum_{j\in H} r_j \cdot c_j(T)\right] > \mathsf{err}^\star$$

then there exists a set $W \subseteq S$ of size at least $\mu \cdot |\mathcal{L}_X|$ 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 $\mu \cdot |\mathcal{L}_X|$ rows $\hat{r}_i$ in W has at least $\mu \cdot |\mathcal{L}_Y^{(i)}| = \mu \cdot |D|/|\mathcal{L}_X|$ 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.

4.2.2.4 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 $D := \cup_{i\in\mathcal{L}_X}(\{i\} \times \mathcal{L}_Y^{(i)})$) on which we test proximity.

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

6. The verifier samples a random $j \leftarrow \mathcal{L}_Y^{(i)}$, 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 $2\mu + \mathsf{err}^\star$. 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 $\mathcal{L}_Y^{(i)}$ (the points on which they coincide), and let $T \subseteq S$ be the rows in S that also have $\hat{v}(i) = \sum_{j \in H} r_j \cdot \hat{r}_i(j)$.

We bound the probability that the verifier rejects when T is small: we argue that if $|T| < \mu \cdot |\mathcal{L}_X|$ then the verifier accepts with probability at most 2μ. The verifier samples $i \leftarrow \mathcal{L}_X$. With probability at most μ, it holds that $i \in T$. If $i \notin T$, either $\hat{v}(i) \neq \sum_{j \in H} r_j \cdot \hat{r}_i(j)$, 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 $\mathcal{L}_Y^{(i)}$, 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 $2\mu + \mathsf{err}^\star$, it follows that $|T| \geq \mu \cdot |\mathcal{L}_X|$ with probability greater than $\mathsf{err}^\star$. Consequently, by Theorem 8, there exists $W \subseteq S$ with $|W| \geq \mu \cdot |\mathcal{L}_X|$ where the columns $(c_j)_{j \in H}$ defined by the rows $(\hat{r}_i)_{i \in \mathcal{L}_X}$ 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) $2\sqrt{1-\delta} + \mathsf{err}^\star$. See Section 4.5.2 for a formal statement and proof. In order to achieve our IOPP for univariate polynomials (Theorem 5), 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.

4.2.3 Testing RS codes with inverse polynomial error

We combine the tools developed in Section 4.2.1 and Section 4.2.2 in order to prove Theorem 5.

Let $f \colon \mathcal{L} \to \mathbb{F}$ be a function claimed to be in $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$. We recursively reduce the degree by a square root until it is a constant. Proximity to $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, O(1)]$ 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 $\mathcal{L}$ is a multiplicative subgroup of $\mathbb{F}^*$. In step i of the recursion, let $d_i = 2^{2^{t-i}} - 1$ and $\mathcal{L}_i$ be an evaluation domain chosen such that $\mathsf{RS}'[\mathbb{F}, \mathcal{L}_i, d_i]$ has rate $\rho_i = (d_i + 1)/|\mathcal{L}_i| = (d+1)/|\mathcal{L}| = \rho$. Assume that there exists a low-degree test for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_{i+1}]$. According to Lemma 3 there exists a poly-IOPP for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_i]$ where the prover's messages are degree d_{i+1} polynomials. We then combine the poly-IOPP with the proximity test for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_{i+1}]$ using the transformation of Lemma 1, whose output is a standard IOPP for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_i]$. The soundness error of the resulting IOPP is (roughly) $2\sqrt{1-\delta} + \beta_{\mathsf{prx}}(1 - \rho^{1/\Omega(1)}) + \frac{\mathrm{poly}(|\mathcal{L}|)}{|\mathbb{F}|}$, where $\beta_{\mathsf{prx}}(1 - \rho^{1/\Omega(1)})$

is the soundness error of the proximity test for $\mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_{i+1}]$ for $(1-\rho^{1/\Omega(1)})$-far functions.

There are $t = \operatorname{loglog} d$ levels of recursion. In each layer, the soundness error is increased by (very roughly) a $\rho^{1/\Omega(1)} + \operatorname{poly}(|\mathcal{L}|)/|\mathbb{F}|$ factor. Hence, the soundness error of the resulting IOPP is $2\sqrt{1-\delta} + \operatorname{loglog} d \cdot (\rho^{1/\Omega(1)} + \operatorname{poly}(|\mathcal{L}|)/|\mathbb{F}|)$. In a more general and tight analysis we can get the soundness error down to

$$\max\left\{2\sqrt{1-\delta},\ \rho^{1/\Omega(1)}\right\} + \frac{\operatorname{poly}(d, 1/\rho)}{|\mathbb{F}|}.$$

In order to get soundness error $\max\{1-\delta, \rho^{1/\Omega(1)} + \frac{\operatorname{poly}(d,1/\rho)}{|\mathbb{F}|}\}$ and in order to get rid of the assumption that $\mathcal{L}$ is a smooth subgroup as in Theorem 5, we additionally utilize the domain-shifting technique described in Section 4.2.1.2. See Section 4.5 for a formal proof and tight analysis.

This concludes the overview of the proof of Theorem 5.

4.3 Proximity generators for correlated agreement

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

- In Section 4.3.1 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 4.3.2, 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.

4.3.1 Proximity generators

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

Definition 4.3.1. *Let $\mathcal{C} \subseteq \mathbb{F}^\ell$ be a code. We say that $\mathcal{C}$ has a **proximity generator** for ℓ functions with seed length w, proximity bound ψ, error ε, and computation time t, if there exists an t-time computable function $\mathsf{Gen}\colon \{0,1\}^{\mathsf{w}} \to \mathbb{F}^\ell$ such that for every $\delta \in (0, 1-\psi)$ and functions $f_1 \ldots, f_\ell \colon [\ell] \to \mathbb{F}$, if*

$$\Pr\left[\Delta\left(\sum_{i\in[\ell]} r_i \cdot f_i, \mathcal{C}\right) \leq \delta \;\middle|\; \begin{array}{r} \alpha \leftarrow \{0,1\}^{\mathsf{w}} \\ (r_1, \ldots r_\ell) \leftarrow \mathsf{Gen}(\alpha) \end{array}\right] > \varepsilon(\delta)\,,$$

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

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

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

Theorem 4.3.2 ([BCIKS20])**.** *Every Reed–Solomon code $\mathcal{C} = \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ with rate $\rho := (d+1)/|\mathcal{L}|$ has a proximity generator for every $\ell \in \mathbb{N}$ with proximity bound $\psi := \sqrt{\rho}$, computation time $O(\ell)$, and one of the following:*

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

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

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

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

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

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

4.3.2 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 4.3.3. *Let* $\mathcal{C} \subseteq \mathbb{F}^\ell$ *be a code. We say that* $\mathcal{C}$ *has a* **strong *proximity generator*** *for* ℓ *functions with seed length* w*, proximity bound* ψ*, error* ε *and computation time* t*, if there exists a* t*-time computable function* $\mathsf{Gen}\colon \{0,1\}^{\mathsf{w}} \to \mathbb{F}^\ell$ *such that for every* $\delta \in (0, 1-\psi)$*, functions* $f_1 \ldots, f_\ell \colon [\ell] \to \mathbb{F}$ *and every* $S \subseteq [\ell]$ *with* $|S| \geq (1-\delta)\cdot\ell$*, if*

$$\Pr\left[\begin{array}{l} \exists T \subseteq S \\ |T| \geq (1-\delta)\cdot\ell \end{array} \;:\; \exists u \in \mathcal{C},\ u(T) = \textstyle\sum_{j\in[\ell]} r_j \cdot f_j(T) \;\middle|\; \begin{array}{r} \alpha \leftarrow \{0,1\}^{\mathsf{w}} \\ (r_1,\ldots,r_\ell) := \mathsf{Gen}(\alpha) \end{array}\right] > \varepsilon\,.$$

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

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

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

Theorem 4.3.4. *Let* $\mathcal{C} = \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ *be a Reed–Solomon code where* $|\mathbb{F}| \geq |\mathcal{L}|^2$*. Suppose that* $\mathcal{C}$ *has a proximity generator* Gen *for* $\ell+1$ *functions with seed length* w*, proximity bound* ψ*, error* ε*, and computation time* t*.*

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

Theorem 4.3.4, when instantiated with the proximity generators of Theorem 4.3.2, implies that every Reed–Solomon code over a large enough field has a strong proximity generator.

Corollary 4.3.5. *Every Reed–Solomon code* $\mathcal{C} = \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ *with* $\mathbb{F} \geq |\mathcal{L}|^2$ *and rate* $\rho := (d+1)/|\mathcal{L}|$ *has a strong proximity generator for every* $\ell \in \mathbb{N}$ *with proximity bound* $\psi := \sqrt{\rho}$*, computation time* $O(\ell)$*, and one of the following:*

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

Above, ε *is defined as in Theorem 4.3.2.*

Proof of Theorem 4.3.4. Let $\delta \in (0, \psi)$ and let $g\colon \mathcal{L} \to \mathbb{F}$ be the function described in Theorem 4.3.6 where $g(x) = 0$ for every $x \in S$. Since g is zero on S, for any $r_1, \ldots, r_{\ell+1}$:

$$r_{\ell+1} \cdot g(S) + \sum_{i \in [\ell]} r_i \cdot f_i(S) = \sum_{i \in [\ell]} r_i \cdot f_i(S)\,,$$

and so

$$\begin{aligned}
\varepsilon &< \Pr_{(r_1,\ldots,r_\ell) \leftarrow \mathsf{Gen}'(U_{\mathsf{w}})} \left[\begin{array}{ll} \exists T \subseteq S & \\ |T| \geq (1-\delta) \cdot |\mathcal{L}| & : \quad \exists u \in \mathcal{C},\ u(T) = \sum_{j \in [\ell]} r_j \cdot f_j(T) \end{array} \right] \\
&= \Pr_{(r_1,\ldots,r_{\ell+1}) \leftarrow \mathsf{Gen}(U_{\mathsf{w}})} \left[\begin{array}{ll} \exists T \subseteq S & \\ |T| \geq (1-\delta) \cdot |\mathcal{L}| & : \quad \exists u \in \mathcal{C},\ u(T) = r_{\ell+1} \cdot g(T) + \sum_{j \in [\ell]} r_j \cdot f_j(T) \end{array} \right],
\end{aligned}$$

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

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

$$\exists u \in \mathcal{C},\ u(T) = r_{\ell+1} \cdot g(T) + \sum_{j \in [\ell]} r_j \cdot f_j(T)\,,$$

then, by definition, $\Delta\left(r_{\ell+1} \cdot g + \sum_{i \in [\ell]} r_i \cdot f_i, \mathcal{C}\right) \leq \delta$. So:

$$\begin{aligned}
\varepsilon &< \Pr_{(r_1,\ldots,r_{\ell+1}) \leftarrow \mathsf{Gen}(U_{\mathsf{w}})} \left[\begin{array}{ll} \exists T \subseteq S & \\ |T| \geq (1-\delta) \cdot |\mathcal{L}| & : \quad \exists u \in \mathcal{C},\ u(T) = r_{\ell+1} \cdot g(T) + \sum_{j \in [\ell]} r_j \cdot f_j(T) \end{array} \right] \\
&\leq \Pr_{(r_1,\ldots,r_{\ell+1}) \leftarrow \mathsf{Gen}(U_{\mathsf{w}})} \left[\Delta\left(r_{\ell+1} \cdot g + \sum_{i \in [\ell]} r_i \cdot f_i, \mathcal{C} \right) \leq \delta \right].
\end{aligned}$$

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

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

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 $S' \subseteq \mathcal{L}$ where $|S'| \geq (1-\delta) \cdot |\mathcal{L}|$ and $S' \neq S$, it follows that $W \subseteq S$. □

Claim 4.3.6. *Let $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ be a Reed–Solomon code with $|\mathbb{F}| \geq |\mathcal{L}|^2$. For every S with $|S| \geq (1-\delta) \cdot |\mathcal{L}|$ with $\delta \leq 1 - 10d/|\mathcal{L}|$ there exists a function $g\colon \mathcal{L} \to \mathbb{F}$ where:*

- *$g(x) = 0$ for every $x \in S$, and*
- *For every $S' \subseteq \mathcal{L}$ where $|S'| \geq (1-\delta) \cdot |\mathcal{L}|$ 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 $x \in \mathcal{L} \setminus S$, we set $g(x)$ to be a uniformly random element in $\mathbb{F}$.

Fix any set $S' \subseteq \mathcal{L}$ with $k := |S'| \geq (1-\delta) \cdot |\mathcal{L}|$, 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 $|S' \cap S| \leq d$ which means that $|S' \setminus S| \geq |S'| - d \geq 9k/10$ (since $|S'| \geq 10d$ 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 $|\mathcal{C}| \leq |\mathbb{F}|^{d+1}$. For any $10d \leq k \leq |\mathcal{L}|$, the number of sets of size k is at most $\binom{|\mathcal{L}|}{k}$. Thus,

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

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

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

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

4.4 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 4.4.1. *Consider the following ingredients:*

- $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$ *is a poly-IOPP for a relation* $\mathcal{R}$ *where the prover sends* s_{poly} *polynomials where the* i*-th polynomial is of degree* d_i*;*
- $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *is an IOPP for* $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_{\mathsf{max}}]$ *with rate* $\rho_{\mathsf{prx}} := (d_{\mathsf{max}}+1)/|\mathcal{L}_{\mathsf{prx}}|$*;*
- Gen *is proximity generator for* $\mathcal{C}_{\mathsf{prx}}$*,* $2 \cdot s_{\mathsf{poly}}$ *functions, seed length* $\mathsf{w}_\star$*, proximity bound* $\psi_\star$*, error* $\varepsilon_\star$*, and computation time* $\mathsf{t}_\star$*.*

If $\max_{i \in [\mathsf{k}_{\mathsf{poly}}]}\{d_i\} \leq d_{\mathsf{max}}$*,* $0 < \gamma < 1 - \max\{\mathsf{B}, 2\rho_{\mathsf{prx}}\}$*, and* $\mathcal{C}_{\mathsf{prx}}$ *is* (γ, ℓ)*–list decodable, then Theorem 4.4.6 is an IOPP* $(\mathbf{P}, \mathbf{V})$ *for* $\mathcal{R}$ *with the following parameters:*[9]

	poly-IOPP for $\mathcal{R}$	IOPP for $\mathcal{C}_{\mathsf{prx}}$	$\to$ IOPP for $\mathcal{R}$
Notation	$(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$	$(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$	$(\mathbf{P}, \mathbf{V})$
Proximity error	β_{poly}	β_{prx}	$\max\{\beta_{\mathsf{poly}}, \beta_{\mathsf{prx}} + \mathsf{err}^\star + s_{\mathsf{poly}} \cdot d_{\mathsf{max}} \cdot \ell^2/\lvert\mathbb{F}\rvert\}$
Rounds	$\mathsf{k}_{\mathsf{poly}}$	$\mathsf{k}_{\mathsf{prx}}$	$2\mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}} + 1$
Alphabet	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
Proof length	s_{poly}	$\mathsf{l}_{\mathsf{prx}}$	$O(s_{\mathsf{poly}} \cdot \lvert\mathcal{L}_{\mathsf{prx}}\rvert + \mathsf{q}_{\mathsf{poly},\Pi}) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$
Proof queries	$\mathsf{q}_{\mathsf{poly},\Pi}$	$\mathsf{q}_{\mathsf{prx},\Pi}$	$O(\mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\Pi}$
Randomness	$\mathsf{r}_{\mathsf{poly}}$	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{poly}} \cdot \log\lvert\mathbb{F}\rvert + \mathsf{w}_\star + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\mathsf{vt}_{\mathsf{poly}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{poly}} + s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}^2_{\mathsf{poly},\Pi}) + \mathsf{t}_\star + \mathsf{vt}_{\mathsf{prx}}$

Above, $\mathsf{err}^\star := \mathsf{err}^\star(\gamma)$ *and* $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$*.*

This section is organized as follows: (i) in Section 4.4.1 we discuss univariate function quotienting, a crucial tool in our transformation; (ii) in Section 4.4.2 we describe our transformation and discuss its efficiency; and (iii) in Section 4.4.3 we prove its completeness and proximity error.

Remark 4.4.2. If given an poly-IOPP where the verifier queries only $\mathsf{q}_{\mathsf{poly},\ell}$ out of the s_{poly} polynomials, then Theorem 4.4.1 can be improved: Gen is required to be a proximity generator for only $2 \cdot \mathsf{q}_{\mathsf{poly},\ell}$ functions (which may lead to changes in its other parameters), and the parameters of the resulting IOPP are:

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

	poly-IOPP for $\mathcal{R}$	*IOPP for* $\mathcal{C}_{\mathsf{prx}}$	$\to$ *IOPP for* $\mathcal{R}$
Notation	$(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$	$(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$	$(\mathbf{P}, \mathbf{V})$
Proximity error	β_{poly}	β_{prx}	$\max\left\{\beta_{\mathsf{poly}}, \beta_{\mathsf{prx}} + \mathsf{err}^\star + s_{\mathsf{poly}} \cdot d_{\mathsf{max}} \cdot \ell^2 / \lvert\mathbb{F}\rvert\right\}$
Rounds	$\mathsf{k}_{\mathsf{poly}}$	$\mathsf{k}_{\mathsf{prx}}$	$2\mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}} + 1$
Alphabet	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
Proof length	s_{poly}	$\mathsf{l}_{\mathsf{prx}}$	$O(s_{\mathsf{poly}} \cdot \lvert\mathcal{L}_{\mathsf{prx}}\rvert + \mathsf{q}_{\mathsf{poly},\Pi}) + \mathsf{l}_{\mathsf{prx}}$
Oracle input queries	$\mathsf{q}_{\mathsf{poly},y}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{poly},y}$
Proof queries	$\mathsf{q}_{\mathsf{poly},\Pi}$	$\mathsf{q}_{\mathsf{prx},\Pi}$	$O(\mathsf{q}_{\mathsf{poly},\Pi} + \mathsf{q}_{\mathsf{poly},\ell} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\Pi}$
Randomness	$\mathsf{r}_{\mathsf{poly}}$	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{poly}} \cdot \log\lvert\mathbb{F}\rvert + \mathsf{w}_\star + \mathsf{r}_{\mathsf{prx}}$
Verifier running time	$\mathsf{vt}_{\mathsf{poly}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{q}_{\mathsf{poly},\ell} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}^2_{\mathsf{poly},\Pi}) + \mathsf{t}_\star + \mathsf{vt}_{\mathsf{prx}}$

Above, $\mathsf{err}^\star := \mathsf{err}^\star(\gamma)$ and $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$.

4.4.1 Univariate function quotienting

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

Definition 4.4.3. *Let* $f\colon \mathcal{L} \to \mathbb{F}$ *be a function,* $S \subseteq \mathcal{L}$ *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* $\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})\colon \mathcal{L} \to \mathbb{F}$ *is defined follows:*

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

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

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

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 4.4.4. *Let* $\hat{f} \in \mathbb{F}^{\leq d}[X]$ *be a polynomial and* $S \subseteq \mathcal{L}$ *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 4.4.5. *Let $\mathbb{F}$ be a field, $\mathcal{L} \subseteq \mathbb{F}$ be a domain, $f, u \colon \mathcal{L} \to \mathbb{F}$ be functions, $S \subseteq \mathbb{F}$ be a set,* $\mathsf{Ans}, \mathsf{Fill} \colon S \to \mathbb{F}$ *be functions, and set $T := \{x \in S \mid \mathsf{Ans}(x) \neq f(x)\}$. Then:*

$$\Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})) + \frac{|T|}{|\mathcal{L}|} \geq \Delta(\mathsf{Unquotient}(u, S, \mathsf{Ans}), f)\,.$$

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

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

and let $Z := \{z \in \mathcal{L} \mid u(z) = \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(z)\}$. By definition

$$\Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})) = 1 - |Z|/|\mathcal{L}|\,.$$

For every $z \in Z \setminus T$:

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

For every other point $z \in \mathcal{L} \setminus (Z \setminus T)$, it may be that $w(z) \neq f(z)$. Therefore:

$$\Delta(f, w) \leq 1 - \frac{|Z \setminus T|}{|\mathcal{L}|} \leq 1 - \left(\frac{|Z| - |T|}{|\mathcal{L}|}\right) = \Delta(u, \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})) + \frac{|T|}{|\mathcal{L}|}\,.$$

□

4.4.2 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 4.4.6. The (honest) IOPP prover **P** receives as input an instance-witness pair $(\mathbb{x}, \mathbb{w})$, while the IOPP verifier **V** receives as input $\mathbb{x}$ and oracle access to $\mathbb{w}$. They interact as follows.

1. For $i = 1, \ldots, \mathsf{k}_{\mathsf{poly}}$:
   (a) **P**: Compute $\hat{f}_i := \mathbf{P}_{\mathsf{poly}}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i-1}) \in \mathbb{F}^{\leq d_i}[X]$ ($\alpha_1, \ldots, \alpha_{i-1}$ were received in prior rounds). Compute and send $g_i := \hat{f}_i(\mathcal{L}_{\mathsf{prx}}) \in \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_i]$.
   (b) **V**: Sample and send a random field element $x_i \leftarrow \mathbb{F}$.
   (c) **P**: Compute and send $y_i := \hat{f}_i(x_i) \in \mathbb{F}$.
   (d) **V**: Sample and send $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.
2. **P**:

   (a) Compute sets $(Q'_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}$ where Q'_i are the queries by $\mathbf{V}_{\mathsf{poly}}^{\mathbb{w},\hat{f}_1,\ldots,\hat{f}_{\mathsf{k}_{\mathsf{poly}}}}(\mathbb{x},\alpha_1,\ldots,\alpha_{\mathsf{k}_{\mathsf{poly}}})$ to $\hat{f}_i$.
   (b) For every $i \in [\mathsf{k}_{\mathsf{poly}}]$, set $Q_i := Q'_i \cup \{x_i\}$.
   (c) For every $i \in [\mathsf{k}_{\mathsf{poly}}]$, set $\hat{f}_i^Q := \mathsf{PolyQuotient}(\hat{f}_i, Q_i) \in \mathbb{F}^{\leq d_i - |Q_i|}$.
   (d) Send $((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{poly}}]}$ 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 $i \in [\mathsf{k}_{\mathsf{poly}}]$, set the degree $e_i := d_{\mathsf{max}} - (d_i - |Q_i|)$.
   (b) Sample and send $\alpha \leftarrow \{0,1\}^{\mathsf{w}_\star}$. Set $(r_1,\ldots,r_{2\mathsf{k}_{\mathsf{poly}}}) := \mathsf{Gen}(\alpha)$.

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

$$\forall\, x \in \mathcal{L}_{\mathsf{prx}}\,,\ u(x) := \sum_{i\in[\mathsf{k}_{\mathsf{poly}}]} r_i \cdot h_i(x) + r_{2i} \cdot x^{e_i} \cdot h_i(x)\,.$$

   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) $\mathbf{V}_{\mathsf{poly}}^{\mathbb{w},\hat{f}_1,\ldots,\hat{f}_{\mathsf{k}_{\mathsf{poly}}}}(\mathbb{x},\alpha_1,\ldots,\alpha_{\mathsf{k}_{\mathsf{poly}}}) = 1$, 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 $i \in [\mathsf{k}_{\mathsf{poly}}]$, $\mathsf{Ans}_i(x_i) = y_i$.
   (c) For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $t \in Q_i \cap \mathcal{L}_{\mathsf{prx}}$ check that $g_i(t) = \mathsf{Ans}_i(t)$ (by querying $g_i(t)$).
   (d) $\mathbf{V}_{\mathsf{prx}}$ 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 $i \in [\mathsf{k}_{\mathsf{poly}}]$, query g_i at t;
      ii. for every $i \in [\mathsf{k}_{\mathsf{poly}}]$, compute $\ell_i := \mathsf{Quotient}(g_i, Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i)(t)$;
      iii. return the value $u(t) := \sum_{i\in[\mathsf{k}_{\mathsf{poly}}]} r_i \cdot \ell_i + r_{2i} \cdot t^{e_i} \cdot \ell_i$.

Remark 4.4.7. 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 $2 \cdot s_{\mathsf{poly}}$ functions. As a result, the coefficients generated by it are indexed until $2s_{\mathsf{poly}}$, rather than $2\mathsf{k}_{\mathsf{poly}}$.
- 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 s_{poly} 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 $\mathsf{k}_{\mathsf{poly}}$-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 $2\mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}} + 1$ rounds.

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

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

- *Proof queries.* The new IOPP verifier queries $(y_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}$ and the list $((Q_i, \mathsf{Ans}_i, \mathsf{Fill}_i))_{i\in[\mathsf{k}_{\mathsf{poly}}]}$, and, for every i, at most $|Q_i|$ queries to g_i in Item 5c (for a total of $\mathsf{q}_{\mathsf{poly},\Pi}$ overall from this item). Moreover, each oracle input query of the low-degree test yields $\mathsf{k}_{\mathsf{poly}}$ queries, and each proof query remains one proof query. The query complexity is therefore $O(\mathsf{q}_{\mathsf{poly},\Pi} + \mathsf{k}_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\Pi}$. (This value changes to $O(\mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f}) + \mathsf{q}_{\mathsf{prx},\Pi}$ if the poly-IOPP prover sends multiple polynomials per round, and s_{poly} polynomials in total.)

- *Randomness.* The new IOPP verifier uses at most $\mathsf{r}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{poly}} \cdot \log|\mathbb{F}| + \mathsf{w}_\star + \mathsf{r}_{\mathsf{prx}}$ bits of randomness. (This value does not change if the poly-IOPP prover sends multiple polynomials per round, as 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 $O(|Q_i|^2)$ order to compute the quotients. The verifier therefore runs in time $O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}^2_{\mathsf{poly},\Pi}) + \mathsf{t}_\star + \mathsf{vt}_{\mathsf{prx}}$. (This value changes to $O(\mathsf{vt}_{\mathsf{poly}} + s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}^2_{\mathsf{poly},\Pi}) + \mathsf{t}_\star + \mathsf{vt}_{\mathsf{prx}}$ if the poly-IOPP prover sends multiple polynomials in a single round since the verifier must query every function in the proximity test.)

4.4.3 Completeness and soundness

We prove Theorem 4.4.1. We first analyze completeness and then soundness.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. 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:

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

where $\mathsf{tr}_{\mathsf{prx}}$ is a transcript of $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$. Let $(r_1, \ldots, r_{2\mathsf{k}_{\mathsf{poly}}}) := \mathsf{Gen}(\alpha)$.

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

$$\mathbf{V}_{\mathsf{poly}}^{\mathbb{w}, \hat{f}_1, \ldots, \hat{f}_{\mathsf{k}_{\mathsf{poly}}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{poly}}}) = 1$$

where $\hat{f}_i = \mathbf{P}_{\mathsf{poly}}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_i)$ is a polynomial of degree (at most) d_i; moreover, in this execution $\mathbf{V}_{\mathsf{poly}}$ queries $\hat{f}_i$ at a set $Q_i' \subseteq Q_i$.

Since $\mathbf{P}$ sends $\mathsf{Ans}_i = \hat{f}_i(Q_i)$, $\mathbf{V}$ answers every query of $\mathbf{V}_{\mathsf{poly}}$ consistently with the polynomials $\hat{f}_1, \ldots, \hat{f}_{\mathsf{k}_{\mathsf{poly}}}$, which means that $\mathbf{V}$ does not reject in Item 5a. Moreover, $\mathbf{V}$ does not reject in Item 5b and Item 5c either since for every i: (a) $\mathsf{Ans}_i(x_i) = \hat{f}_i(x_i) = y_i$ and, (b) $g_i(t) = \hat{f}_i(t) = \mathsf{Ans}_i(t)$ for every $t \in Q_i \cap \mathcal{L}_{\mathsf{prx}}$. We are left to argue that $\mathbf{V}$ does not reject in Item 5d, 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 $e_i := d_{\mathsf{max}} - (d_i - |Q_i|)$, both $\hat{h}_i$ and $X^{e_i} \cdot \hat{h}_i$ have degree at most d_{max}. Consequently, for any coefficients $r_1, \ldots, r_{2\mathsf{k}_{\mathsf{poly}}}$, the following linear combination has degree at most d_{max}:

$$\hat{u}(X) := \sum_{j \in [\mathsf{k}_{\mathsf{poly}}]} r_i \cdot \hat{h}_i(X) + r_{2i} \cdot X^{e_i} \cdot \hat{h}_i(X).$$

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

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

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

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

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

Let $(r_1, \ldots, r_{2\mathsf{k}_{\mathsf{poly}}}) := \mathsf{Gen}(\alpha)$ and, for every $i \in [\mathsf{k}_{\mathsf{poly}}]$, 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 $\Delta(g_i, \hat{w}_i(\mathcal{L}_{\mathsf{prx}})) \leq \gamma$ 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 $\Delta(h_i, \hat{v}_i(\mathcal{L}_{\mathsf{prx}})) \leq \gamma$.

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 $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$.

Claim 4.4.8. $\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1 \mid \mathsf{E}_{\mathsf{prx}} \ \wedge \ \mathsf{E}_{\mathsf{out}}\ ] \leq \beta_{\mathsf{poly}}$.

Proof. Suppose towards contradiction that the above probability is greater than β_{poly}. Below we construct a prover $\tilde{\mathbf{P}}_{\mathsf{poly}}$ that convinces $\mathbf{V}_{\mathsf{poly}}$ on $\mathbb{x}$ with probability greater than β_{poly}, contradicting the soundness property of $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$. Prior to round $i \in [\mathsf{k}_{\mathsf{poly}}]$, $\tilde{\mathbf{P}}_{\mathsf{poly}}$ has chosen (and stored) points $x_1, \ldots, x_{i-1}$ and has received the messages $\alpha_1, \ldots, \alpha_{i-1}$ from the verifier.

$\tilde{\mathbf{P}}_{\mathsf{poly}}$ in round $i \in [\mathsf{k}_{\mathsf{poly}}]$:
1. Compute $g_i := \tilde{\mathbf{P}}(x_1, \alpha_1, \ldots, x_{i-1}, \alpha_{i-1})$.
2. Sample $x_i \leftarrow \mathbb{F}$. (Pass this value on as state for the next round.)
3. Compute $y_i := \tilde{\mathbf{P}}(x_1, \alpha_1, \ldots, x_{i-1}, \alpha_{i-1}, x_i)$. 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 $\Delta(\hat{w}_i(\mathcal{L}_{\mathsf{prx}}), g_i) \leq \gamma$ then send $\hat{w}_i$. Otherwise abort.
4. Receive α_i from $\mathbf{V}_{\mathsf{poly}}$.

We analyze the probability that $\tilde{\mathbf{P}}_{\mathsf{poly}}$ convinces $\mathbf{V}_{\mathsf{poly}}$. 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 $\Delta(\hat{v}_i(\mathcal{L}_{\mathsf{prx}}), h_i) \leq \gamma$. 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 Theorem 4.4.5:

$$\Delta(\hat{w}_i(\mathcal{L}_{\mathsf{prx}}), g_i) \leq \Delta(\hat{v}_i(\mathcal{L}_{\mathsf{prx}}), h_i) + |T_i|/|\mathcal{L}_{\mathsf{prx}}| \leq \gamma + |T_i|/|\mathcal{L}_{\mathsf{prx}}| ,$$

where T_i is the set of points $t \in Q_i \cap \mathcal{L}_{\mathsf{prx}}$ where $g_i(t) \neq \mathsf{Ans}_i(t)$.

If $\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}$ holds and $\mathbf{V}$ accepts then, by Item 5c, for every $t \in Q_i \cap \mathcal{L}_{\mathsf{prx}}$ it holds that $g_i(t) = \mathsf{Ans}_i(t)$, and so $|T_i| = 0$. Moreover, $\hat{w}_i(a) = \mathsf{Ans}_i(a)$ 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 $\tilde{\mathbf{P}}_{\mathsf{poly}}$ sends $\hat{w}_i$ to $\mathbf{V}_{\mathsf{poly}}$.

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

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

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

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

Claim 4.4.9. $\Pr[\ \neg \mathsf{E}_{\mathsf{out}}\ ] \leq \mathsf{k}_{\mathsf{poly}} \cdot d_{\mathsf{max}} \cdot \ell^2 / |\mathbb{F}|$. *(If the prover sends s_{poly} messages overall, then this is amended to:* $\Pr[\ \neg \mathsf{E}_{\mathsf{out}}\ ] \leq s_{\mathsf{poly}} \cdot d_{\mathsf{max}} \cdot \ell^2 / |\mathbb{F}|$.*)*

Proof. For every i, $\mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_i]$ is (γ, ℓ)-list decodable: $\mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_i] \subseteq \mathcal{C}_{\mathsf{prx}}$ and $\mathcal{C}_{\mathsf{prx}}$ is (γ, ℓ)-list decodable (see Theorem 2.2.2).

Fix $i \in [\mathsf{k}_{\mathsf{poly}}]$ and consider $g_i \colon \mathcal{L}_{\mathsf{prx}} \to \mathbb{F}$ sent by $\tilde{\mathbf{P}}_{\mathsf{poly}}$. By the polynomial identity lemma (Theorem 2.7.1), 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 $\mathcal{L}_{\mathsf{prx}}$. 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 $i \in [\mathsf{k}_{\mathsf{poly}}]$. □

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

Claim 4.4.10. $\Pr[\ \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{w}} \rangle(\mathbb{x}) = 1 \mid \neg \mathsf{E}_{\mathsf{prx}}\ ] \leq \mathsf{err}^\star(\gamma) + \beta_{\mathsf{prx}}$.

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

$$\Pr[\Delta(u, \mathcal{C}_{\mathsf{prx}}) \leq \gamma\ ] \leq \mathsf{err}^\star(\gamma)\,. \tag{4.1}$$

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 $\mathsf{err}^\star(\gamma) + \beta_{\mathsf{prx}}$. We now show that Equation 4.1 holds.

Suppose towards contradiction that

$$\Pr[\Delta(u, \mathcal{C}_{\mathsf{prx}}) \leq \gamma] > \mathsf{err}^\star(\gamma)\,. \tag{4.2}$$

Recall that

$$u(x) := \sum_{j \in [\mathsf{k}_{\mathsf{poly}}]} r_i \cdot h_i(x) + r_{2i} \cdot x^{e_i} \cdot h_i(x)\,,$$

where $(r_1, \ldots, r_{2\mathsf{k}_{\mathsf{poly}}}) := \mathsf{Gen}(\alpha)$ for uniformly random seed α. Observe that $0 < \gamma < 1 - \mathsf{B}$, and that Gen is a proximity generator for $2 \cdot \mathsf{k}_{\mathsf{poly}}$ with proximity bound $\psi_\star$

and error $\varepsilon_\star$. Therefore, as a result of Equation 4.2, it holds that h_i and $X^{e_i} \cdot h_i$ have correlated agreement with the code $\mathcal{C}_{\mathsf{prx}}$ on a set S of size $|S| \geq (1-\gamma) \cdot |\mathcal{L}_{\mathsf{prx}}|$. Let p and q be the closest codewords that agree with h_i and $X^{e_i} \cdot h_i$ 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 $\hat{p}'_i(X) := X^{e_i} \cdot \hat{p}(X)$.

We now reach a contradiction by showing that it simultaneously holds that

$$\Delta(\hat{p}'_i(\mathcal{L}_{\mathsf{prx}}), \hat{q}(\mathcal{L}_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|\,,$$

and that $\Delta(\hat{p}'_i(\mathcal{L}_{\mathsf{prx}}), \hat{q}(\mathcal{L}_{\mathsf{prx}})) < 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|$:

- $\Delta(\hat{p}'_i(\mathcal{L}_{\mathsf{prx}}), \hat{q}(\mathcal{L}_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|$: since every polynomial $\hat{v}_i \in \mathbb{F}^{\leq d_i - |Q_i|}[X]$ has $\Delta(\hat{v}_i(\mathcal{L}_{\mathsf{prx}}), h_i) > \gamma$, and $\Delta(\hat{p}(\mathcal{L}_{\mathsf{prx}}), h_i) \leq \gamma$, it follows that $d_i - |Q_i| < \deg(\hat{p}) \leq d_{\mathsf{max}}$. Letting $\hat{p}'_i(X) := X^{e_i} \cdot \hat{p}(X)$, we have $d_{\mathsf{max}} < \deg(\hat{p}') \leq 2d_{\mathsf{max}}$. 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 $\Delta(\hat{p}'(\mathcal{L}_{\mathsf{prx}}), \hat{q}(\mathcal{L}_{\mathsf{prx}})) \geq 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|$.
- $\Delta(\hat{p}'_i(\mathcal{L}_{\mathsf{prx}}), \hat{q}(\mathcal{L}_{\mathsf{prx}})) < 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|$: This follows from the observation that $X^{e_i} \cdot \hat{p}$ and $\hat{q}$ agree on S, where $|S| \geq (1-\gamma) \cdot |\mathcal{L}_{\mathsf{prx}}|$ for $\gamma < 1 - 2\rho_{\mathsf{prx}} < 1 - 2d_{\mathsf{max}}/|\mathcal{L}_{\mathsf{prx}}|$.

As we reached a contradiction, we can conclude that Equation 4.1 holds, thereby proving the claim. □

Define $p := \Pr[\mathsf{E}_{\mathsf{prx}} \wedge \mathsf{E}_{\mathsf{out}}]$. Putting together Theorem 4.4.8, Theorem 4.4.9, and Theorem 4.4.10, we obtain:

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

If the prover sends s_{poly} messages overall, then this is amended to:

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

4.5 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 4.5.1. *Let $\mathbb{F}$ be a field, G be a multiplicative subgroup of $\mathbb{F}^*$ whose order is a power of two, and $\mathcal{L} \subseteq \mathbb{F}$ be a set.*

If $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot |\mathcal{L}|$ then the Reed–Solomon code $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ 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	$O(\lvert\mathcal{L}\rvert/\rho)$
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, $\rho := (d+1)/|\mathcal{L}|$ is the rate of $\mathcal{C}$.

This section is organized in four parts.

- In Section 4.5.1 we describe a univariate sumcheck that we use as a subroutine.
- In Section 4.5.2 we construct a poly-IOPP for bivariate Reed–Muller codes.
- In Section 4.5.3 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 4.5.4 we use the compiler in Section 4.4 to recursively transform the poly-IOPP from Section 4.5.3 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.

4.5.1 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 4.5.2. *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 γ be a claimed sum. The protocol $(\mathbf{P}_\Sigma, \mathbf{V}_\Sigma)$ in Theorem 4.5.3 satisfies the following properties.*

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

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

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

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

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* $O(\log|H|)+\mathrm{t}_w$ *(field operations) where* t_w *is the time to evaluate* $\hat{w}$ *on a random field element.*

Construction 4.5.3.

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

Proof of Theorem 4.5.2. We prove completeness and then soundness. We rely on the following fact.

Fact 4.5.4. *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 $\sum_{\alpha\in H}\hat{w}(\alpha)\cdot\hat{f}(\alpha)=\gamma$. 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 Theorem 4.5.4, $\sum_{\alpha\in H}\hat{q}(\alpha)=\hat{q}(0)\cdot|H|$. Since $\gamma=\sum_{\alpha\in H}\hat{w}(\alpha)\cdot\hat{f}(\alpha)=\sum_{\alpha\in H}\hat{q}(\alpha)$, we can write:
$$\hat{w}(X)\cdot\hat{f}(X)=\hat{h}(X)\cdot\hat{V}_H(X)+(X\cdot\hat{p}(X)+\gamma/|H|)\,,$$
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
$$\hat{w}(a)\cdot f(a)=h(a)\cdot\hat{V}_H(a)+(a\cdot p(a)+\gamma/|H|)\,.$$

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

Soundness. Suppose that $\sum_{\alpha\in H} \hat{w}(\alpha)\cdot\hat{f}(\alpha) \neq \gamma$ 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:

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

(This holds since, by using Theorem 4.5.4, otherwise $\sum_{\alpha\in H} \hat{w}(\alpha)\cdot\hat{f}(\alpha) = \gamma$.) Since both polynomials have degree at most $d_f + d_w$, by the polynomial identity lemma (Theorem 2.7.1):

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

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

4.5.2 poly-IOPP for bivariate RM codes

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

Definition 4.5.5. *A domain $D \subseteq \mathbb{F}\times\mathbb{F}$ is* **admissible** *if there exists a "row-index" domain $\mathcal{L}_X \subseteq \mathbb{F}$ such that $D = \cup_{i\in\mathcal{L}_X}(\{i\}\times\mathcal{L}_Y^{(i)})$ where $|\mathcal{L}_Y^{(i)}| = |\mathcal{L}_Y^{(i')}|$ for every $i, i' \in \mathcal{L}_X$.*

Lemma 4.5.6. *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 $\mathcal{L}_X$.*
- *$H \subseteq \mathbb{F}$ is a multiplicative subgroup of $\mathbb{F}^*$.*
- *Gen is a strong proximity generator for $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, d_X]$ for $|H|$ functions with seed length $\mathsf{w}_\star$, proximity bound $\psi_\star$, and error $\varepsilon_\star$.*

If $|\mathbb{F}| \geq |\mathcal{L}_X|^2$ and $|H| > d_Y$, then Theorem 4.5.7 is a poly-IOPP for $\mathcal{C}$ with the following parameters:

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

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

Construction 4.5.7. 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 **P** receives as input f, whereas the verifier **V** is given oracle access to f.

1. **P**: Compute $\hat{f}$ and then, for every $i \in \mathcal{L}_X$, send $\hat{r}_i(X) := \hat{f}(i, X) \in \mathbb{F}^{\leq d_Y}[X]$.
2. **V**: Sample and send $\alpha \leftarrow \{0,1\}^{\mathsf{w}_\star}$.
3. **P**: Let $(r_\ell)_{\ell\in H} \leftarrow \mathsf{Gen}(\alpha)$ (formally, we associate each $x \in H$ with a unique index in $[|H|]$). Interpolate the points of $\vec{r}$ to define the weight polynomial $\hat{w} \in \mathbb{F}^{\leq |H|-1}[X]$ such that $\hat{w}(j) = r_j$ 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. **V**: Sample $i \leftarrow \mathcal{L}_X$ uniformly at random and send to **P**.
5. **P** and **V** run the interaction phase of the weighted univariate sumcheck protocol $(\mathbf{P}_\Sigma, \mathbf{V}_\Sigma)$ (as in Theorem 4.5.2) 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 $j \leftarrow \mathcal{L}_Y^{(i)} := \{j \in \mathbb{F} \mid (i,j) \in D\}$ 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 $|\mathcal{L}_X|$ different polynomials $(\hat{r}_i)_{i\in\mathcal{L}_X}$. It then sends a polynomial v and, in the univariate sumcheck, sends two additional polynomials. Thus the prover sends a total of $|\mathcal{L}_X| + 3$ 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 α using $\mathsf{w}_\star$ 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 $3\log|\mathbb{F}| + \mathsf{w}_\star$.
- *Verifier running time.* The verifier chooses α, 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 α since α is the input to Gen. The running time of the verifier is, therefore $O(\log|H| + \widehat{\mathsf{t}}_{\mathsf{prx}})$.

Proof of Theorem 4.5.6. We prove completeness and then proximity.

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

$$\mathsf{tr} = \left((\hat{r}_i)_{i\in\mathcal{L}_X}, \alpha, \hat{v}, (i,j), \mathsf{tr}_\Sigma\right) ,$$

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

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 $i \in \mathcal{L}_X$, where $\hat{w}$ is the extension of the coefficients $(r_\ell)_{\ell\in H}$ 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 $i \in \mathcal{L}_X$, 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:

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

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 $\delta < 1 - \mathsf{B}^2$. Set $\mu := \sqrt{1-\delta}$ and suppose that

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

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

A transcript of the protocol has the following structure:

$$\mathsf{tr} = \left((\hat{r}_i)_{i\in\mathcal{L}_X}, \alpha, \hat{v}, (i,j), \mathsf{tr}_\Sigma\right) ,$$

where $(\hat{r}_i)_{i\in\mathcal{L}_X}$ 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 $(r_\ell)_{\ell\in H} := \mathsf{Gen}(\alpha)$ and let $c_1,\ldots,c_{|H|}\colon \mathcal{L}_X \to \mathbb{F}$ 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:

$$S := \left\{ i \in \mathcal{L}_X \,\middle|\, |\{j \in \mathcal{L}_Y^{(i)} \;:\; \hat{r}_i(j) = f(i,j)\}| \geq \mu \cdot |\mathcal{L}_Y^{(i)}| \right\},$$

and

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

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

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

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

Claim 4.5.8. $\Pr[\ \langle \mathbf{P},\mathbf{V}\rangle = 1 \ \wedge\ |T| < \mu\cdot|\mathcal{L}_X|\ ] \leq \mu + \max\left\{\mu, \frac{d_Y+|H|-1}{|\mathbb{F}|}\right\}$.

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.

- $\hat{v}(i) \neq \sum_{j\in H} r_j \cdot \hat{r}_i(j)$: 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 $(d_Y + |H| - 1)/|\mathbb{F}|$.
- $|\{j \in \mathcal{L}_Y^{(i)} \;:\; \hat{r}_i(j) = f(i,j)\}| < \mu\cdot|\mathcal{L}_Y^{(i)}|$: 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

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

□

It follows from Theorem 4.5.8 that,

$$\Pr[\ |T| \geq \mu\cdot|\mathcal{L}_X|\ ] \geq \Pr[\ \langle \mathbf{P},\mathbf{V}\rangle = 1 \ \wedge\ |T| \geq \mu\cdot|\mathcal{L}_X|\ ] \geq \mathsf{err}^*(1-\mu)\,.$$

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

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

Since $0 < \delta < 1 - \psi_\star^2$ and $\delta := 1 - \mu^2$, it follows that $0 < 1 - \mu < 1 - \psi_\star$. Since Gen is a strong proximity generator with proximity bound B and error $\varepsilon_\star$, we conclude that there exists a set $W \subseteq S$ with $|W| \geq \mu \cdot |\mathcal{L}_X|$ 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 4.5.9. *There exists a bivariate polynomial $\hat{Q} \in \mathbb{F}[X,Y]$ with $\deg_X(\hat{Q}) \leq d_X$ and $\deg_Y(\hat{Q}) \leq d_Y$ such that $\hat{Q}(i,j) = \hat{r}_i(j)$ for every (i,j) where $i \in W$ and $j \in \mathcal{L}_Y^{(i)}$.*

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 $\deg_X(\hat{Q}) \leq d_X$, that $\deg_Y(\hat{Q}) \leq d_Y$, and that $\hat{Q}(i,j) = r_i(j) = \hat{c}_j(i)$ for every $(i,j) \in W' \times H$. Since $\deg_X(\hat{Q}), \deg(\hat{c}_j) \leq d_X$ 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 $|H| > d_Y$ points. Therefore, $\hat{Q}(i,j) = \hat{r}_i(j)$ for every (i,j) where $i \in W$ and $j \in \mathcal{L}_Y^{(i)}$. □

Finally, by applying Theorem 4.5.9 and recalling that $i \in W \subseteq S$ only if at least a μ-fraction of the values $j \in \mathcal{L}_Y^{(i)}$ 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, $\Delta(f,\mathcal{C}) \leq \Delta(f,\hat{Q}) \leq 1 - \mu^2$. □

4.5.3 poly-IOPP for RS codes

We construct a poly-IOPP for Reed–Solomon codes. In more detail, we show that Theorem 4.5.6 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 Theorem 4.5.6.

Claim 4.5.10 ([BS08]). *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 $\deg_X(\hat{Q}) = \left\lfloor \deg(\hat{f})/\deg(\hat{q}) \right\rfloor$ and $\deg_Y(\hat{Q}) \leq \deg(\hat{q}) - 1$ 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 $\deg(\hat{f}) \leq t \cdot \deg(\hat{q}) - 1$ then $\deg_X(\hat{Q}) \leq t - 1$.*
- *For every $\hat{Q} \in \mathbb{F}[X,Y]$ with $\deg_X(\hat{Q}) \leq t - 1$ and $\deg_Y(\hat{Q}) \leq \deg(\hat{q}) - 1$, the polynomial $\hat{f}(Z) := \hat{Q}(\hat{q}(Z), Z)$ has degree $\deg(\hat{f}) \leq t \cdot \deg(\hat{q}) - 1$.*

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

Lemma 4.5.11. *Consider the following ingredients:*

- *$\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ 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 $D := \{(\hat{q}(j), j) \mid j \in \mathcal{L}\}$ is an admissible domain with row-index domain $\mathcal{L}_X = \{\hat{q}(j) \mid j \in \mathcal{L}\}$.*
- *$H \subseteq \mathbb{F}$ is a multiplicative subgroup of $\mathbb{F}^*$.*
- *Gen is a strong proximity generator for $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, t-1]$ for $|H|$ functions with seed length $\mathsf{w}_\star$, proximity bound $\psi_\star$, and error $\varepsilon_\star$.*

If $|\mathbb{F}| \geq |\mathcal{L}_X|^2$ and $|H| > d_q - 1$ then Theorem 4.5.12 is a poly-IOPP for $\mathcal{C}$ with the following parameters:

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

Above, $\varepsilon_\star := \varepsilon_\star(1-\sqrt{1-\delta})$ *and* $\widehat{\mathsf{t}}_\star$ *is defined as in Theorem 4.5.6.*

Construction 4.5.12. 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 $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$ described in Theorem 4.5.6 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 $\mathbf{V}_{\mathsf{poly}}$ to f', $\mathbf{V}$ queries $f(t)$ and returns the answer to $\mathbf{V}_{\mathsf{poly}}$.

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

Proof. We prove completeness and proximity.

Completeness. If $f \in \mathcal{C}$ then, by Theorem 4.5.10, there exists a polynomial $\hat{Q}$ with $\deg_X(\hat{Q}) \leq d_X$ and $\deg_Y(\hat{Q}) \leq d_Y$ 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 Theorem 4.5.6, it follows that $\mathbf{V}_{\mathsf{poly}}$ accepts with probability 1. Consequently, $\mathbf{V}$ always accepts.

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

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

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 Theorem 4.5.10, $\deg(\hat{p}) \leq t \cdot d_q - 1 = d_f$, 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: $\Delta(f, \mathcal{C}) \leq \Delta(f, \hat{p}(\mathcal{L})) \leq \delta$. □

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

Corollary 4.5.13. *Consider the following ingredients:*

- $\mathbb{F}$ *is a field,* $\mathcal{L}$ *and* H *are multiplicative subgroups of* $\mathbb{F}^*$ *where* $|H|$ *divides* $|\mathcal{L}|$.
- Gen *is a strong proximity generator for* $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, t-1]$ *for* $|H|$ *functions with seed length* $\mathsf{w}_\star$*, proximity bound* $\psi_\star$*, and error* $\varepsilon_\star$*, where* $\mathcal{L}_X := \{a^{|H|} \mid a \in \mathcal{L}\}$ *and* $t \in \mathbb{N}$.

If $|\mathbb{F}| \geq \left(\frac{|\mathcal{L}|}{|H|}\right)^2$ *then there is a poly-IOPP for* $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, t \cdot |H| - 1]$ *with the following parameters:*

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

Above, $\varepsilon_\star := \varepsilon_\star(1 - \sqrt{1-\delta})$ and $\widehat{\mathsf{t}}_\star$ is defined as in Theorem 4.5.6.

Proof. We use the poly-IOPP described in Theorem 4.5.11 using $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, t \cdot |\mathcal{L}| - 1]$, $\hat{q}(X) := X^{|H|}$, the multiplicative subgroup H, and the proximity generator Gen.

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

Finally, observe that $|\mathbb{F}| > \left(\frac{|\mathcal{L}|}{|H|}\right)^2 = |\mathcal{L}_X|^2$ and $|H| = \deg(\hat{q}) > \deg(\hat{q}) - 1$, and so all of the requirements for Theorem 4.5.11 hold. The parameters follow. □

4.5.4 Recursive construction of IOPP for RS codes

We now prove Theorem 4.5.1 showing an IOP of proximity for the Reed–Solomon code that is compatible with the inverse-polynomial soundness error regime.

4.5.4.1 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 4.5.14. *Consider the following ingredients:*

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

If $|\mathbb{F}| \geq \left(\frac{|\mathcal{L}|}{|H|}\right)^2$ and $d_{\mathsf{prx}} \geq |H| - 1$, then there is an IOPP for $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, |H|^2 - 1]$ 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	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert\mathcal{L}_{\mathsf{prx}}\rvert \cdot \lvert H\rvert/\rho) + \mathsf{l}_{\mathsf{prx}}$
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, $\rho := |H|^2/|\mathcal{L}|$, $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}}+1)/|\mathcal{L}_{\mathsf{prx}}|$, *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 $\mathcal{L}_X := \{a^{|H|} \mid a \in \mathcal{L}\}$ and $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, |H|-1]$. Observe that $|\mathcal{L}_X| = |\mathcal{L}|/|H|$ and that the rate of $\mathcal{C}_X$ is $\rho_X := |H|/|\mathcal{L}_X| = |H|^2/|\mathcal{L}| = \rho$ where ρ is the rate of $\mathcal{C}$.

Consider the poly-IOPP for $\mathcal{C}$ described in Theorem 4.5.13 with $t := |H|$ using the following ingredients:

- The field $\mathbb{F}$ and multiplicative subgroups $\mathcal{L}$ and H (observe that $|H|$ divides $|\mathcal{L}|$).
- Gen is the strong proximity generator for $\mathcal{C}_X$, described in Item 1 of Theorem 4.3.5 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 $\widehat{\mathsf{t}}_\star = \tilde{O}(|H|)$.[10]

Observe that $|\mathbb{F}| \geq \left(\frac{|\mathcal{L}|}{|H|}\right)^2$ and so, as a result, the requirements for Theorem 4.5.13 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	$2\sqrt{1-\delta} + O\left(\frac{\lvert H\rvert^3\cdot(1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)$
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.

[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 compile the poly-IOPP for $\mathcal{C}$ into an IOPP for $\mathcal{C}$ using Theorem 4.4.1 (following Theorem 4.4.2 with $\mathsf{q}_{\mathsf{poly},\ell} := 4$) 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 $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_{\mathsf{prx}}]$ 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 4.3.2 for 8 functions, seed length $8\log|\mathbb{F}|$, proximity bound $\mathsf{B} := \sqrt{\rho_{\mathsf{prx}}}$, 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 $0 < \gamma < 1 - \max\{\mathsf{B}, 2\rho_{\mathsf{prx}}\}$, and, by the Johnson bound Theorem 2.2.5, the code $\mathcal{C}_{\mathsf{prx}}$ is $(\gamma, 1/2\rho_{\mathsf{prx}})$–list decodable. The requirements for Theorem 4.4.1 have therefore been met. The resulting IOPP for $\mathcal{C}$ has proximity error

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

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

	poly-IOPP for $\mathcal{R}$	*IOPP for* $\mathcal{C}_{\mathsf{prx}}$	$\rightarrow$ *IOPP for* $\mathcal{R}$
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*	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert\mathcal{L}_{\mathsf{prx}}\rvert \cdot \lvert H\rvert/\rho) + \mathsf{l}_{\mathsf{prx}}$
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	$\tilde{O}(\lvert H\rvert)$	$\mathsf{vt}_{\mathsf{prx}}$	$\tilde{O}(\lvert H\rvert + \mathsf{q}_{\mathsf{prx},f}) + \mathsf{vt}_{\mathsf{prx}}$

□

We now use Theorem 4.5.14 to recursively construct IOPPs for Reed–Solomon codes for any degree of a specific structure.

Lemma 4.5.15. *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* $\mathcal{L}$ *of order* 2^k.

If $|\mathbb{F}| \geq 2^{2k}$ *and* $k \geq 2m$ *then, letting* $d := 2^{2m} - 1$, *then Theorem 4.5.16 is an IOPP for* $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ *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	$O(\lvert\mathcal{L}\rvert/\rho)$
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 4.5.16. Let g be a generator of $\mathcal{L}$. On input $f\colon \mathcal{L} \to \mathbb{F}$, the protocol is as follows:

1. *If $m < 7$:* **P** and **V** run the low-degree test for constant degree to described in Theorem 2.2.6 to show that f is close to $\mathcal{C}$.

2. *If $m \geq 7$:* Let $d_{\text{prx}} := 2^m - 1$ if m is even, and $d_{\text{prx}} := 2^{m+1} - 1$ otherwise, and let $\mathcal{L}_{\text{prx}} := \langle g^{2^{m-7}} \rangle$. **P** and **V** run the IOPP described in Theorem 4.5.14 to show proximity of f to $\mathcal{C}$, with $H = \langle g^{2^{k-m}} \rangle$ and using an IOPP for $\mathcal{C}_{\text{prx}} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\text{prx}}, d_{\text{prx}}]$ that is instantiated recursively.

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

Basis. When $m < 7$, by Theorem 2.2.6 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 $\mathcal{L}$ be a multiplicative subset of $\mathbb{F}^*$ of order 2^k with $\lvert\mathbb{F}\rvert \geq 2^{2k}$. Define d_{prx}, $\mathcal{L}_{\text{prx}}$ and H as in Item 2 of Theorem 4.5.16 with respect to m and $\mathcal{L}$. Observe the following:

- $\lvert H\rvert = 2^m$ divides $\lvert\mathcal{L}\rvert = 2^k$;
- $d = \lvert H\rvert^2 - 1$;
- $d_{\text{prx}} := 2^{2m'} - 1$ for $m' \in \mathbb{N}$;
- $\lvert\mathcal{L}_{\text{prx}}\rvert = 2^{k-m+7} = O(\lvert\mathcal{L}\rvert/\lvert H\rvert)$;
- $O(\log\log d_{\text{prx}} + 1) = O(\log\log d)$;
- $2^{-7} \cdot \rho \leq \rho_{\text{prx}} \leq 2^{-6} \cdot \rho$.

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

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

Now apply Theorem 4.5.14 using the IOPP for $\mathcal{C}_{\text{prx}}$ as in Item 2. The result of this is an IOPP for $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ with $d := 2^{2m} - 1$ with the following parameters:

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

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

  The proximity error of the resulting IOPP is

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

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

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

  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

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

- ***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 $\tilde{O}(|H|) + \tilde{O}(\sqrt{d_{\mathsf{prx}}}) = \tilde{O}(\sqrt{d})$.

□

4.5.4.2 Proof of Theorem 4.5.1

Let m be the smallest integer such that $d \leq 2^{2m} - 1$ and $\mathcal{L}_{\mathsf{prx}} := G$, and set $d_{\mathsf{prx}} := 2^{2m} - 1$. Observe that $d \leq d_{\mathsf{prx}} < 4d$, that $|\mathcal{L}_{\mathsf{prx}}| \geq 2^8 \cdot |\mathcal{L}|$. We apply Theorem 4.7.2 with the following ingredients:

- The Reed–Solomon code $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$.
- The IOPP for $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_{\mathsf{prx}}]$ described in Theorem 4.5.15. Notice that the rate of $\mathcal{C}_{\mathsf{prx}}$ is $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}} + 1)/|\mathcal{L}_{\mathsf{prx}}| \leq 2^{-6} \cdot \rho$.
- The proximity generator for $\mathcal{C}_{\mathsf{prx}}$ described in Item 2 of Theorem 4.3.2 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 2.2.5, $\mathcal{C}_{\mathsf{prx}}$ is $(\gamma, 1/2\rho_{\mathsf{prx}})$–list decodable. We can therefore apply Theorem 4.7.2 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

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

The proximity error of the resulting IOPP is therefore at most

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

- ***Rounds.*** The number of rounds is $O(\log\log d_{\mathsf{prx}}) + 1 = O(\log\log d)$.
- ***Proof length.*** The proof length is $O\left(|\mathcal{L}_{\mathsf{prx}}| + |\mathcal{L}_{\mathsf{prx}}|/\rho_{\mathsf{prx}}\right) = O(|\mathcal{L}|/\rho)$.
- ***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 $\tilde{O}(\sqrt{d_{\mathsf{prx}}}) + O(1) = \tilde{O}(\sqrt{d})$.

4.6 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 4.6.1.

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

Definition 4.6.1. *The* R1CS *relation* $\mathcal{R}_{\mathsf{R1CS}}$ *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 4.6.2. *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* $\mathcal{R}_{\mathsf{R1CS}}$ *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 Theorem 4.6.3, with Theorem 4.7.1, 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 $O(\text{loglog}\, n)$.
- ***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. □

4.6.1 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 4.6.3. *There exists a poly-IOP for the* NP*-complete relation* R1CS *with the following parameters:*

poly-IOP for R1CS	
Soundness error	$3 \cdot (n-1)/\lvert\mathbb{F}\rvert$
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 4.6.4. 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 $H_{\text{in}} \subseteq H$ be the subset of order $|H_{\text{in}}| = k$ 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 $\hat{V}_{H_{\text{in}}}$ similarly with regards to H_{in}.

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

1. **P**: send the polynomials $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h} \in \mathbb{F}^{\leq |H|-1}[X]$ and $\hat{f}_w \in \mathbb{F}^{\leq |H|-k-1}[X]$ 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) $\hat{f}_w(X) := \frac{\hat{z}(X) - \hat{u}(X)}{\hat{V}_{H_{\mathsf{in}}}(X)}$ where $\hat{z}$ is the unique degree $n-1$ polynomial that is equal to z on H and $\hat{u}$ is the unique degree $k-1$ polynomial that is equal to u on H_{in}.

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 $\hat{q}_{M,r}(X) := \sum_{a \in H} M^\top(X, a) \cdot r^a$ for every $x \in H$.

3. **P** and **V** execute the poly-IOP for univariate sumcheck (Theorem 4.5.2) 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 Theorem 4.6.3. 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:

$$\begin{aligned}
\sum_{\alpha \in H} \hat{p}_r(\alpha) \cdot \hat{f}_M(\alpha) - \hat{q}_{M,r}(\alpha) \cdot \hat{f}(\alpha) &= \sum_{\alpha \in H} r^\alpha \cdot \hat{f}_M(\alpha) - \sum_{\alpha \in H} \sum_{\beta \in H} r^\beta \cdot M^\top(\alpha, \beta) \cdot \hat{f}_z(\alpha) \\
&= \sum_{\alpha \in H} r^\alpha \cdot \hat{f}_M(\alpha) - \sum_{\alpha \in H} \sum_{\beta \in H} r^\alpha \cdot M^\top(\beta, \alpha) \cdot \hat{f}_z(\beta) \\
&= \sum_{\alpha \in H} \left(\hat{f}_M(\alpha) - \sum_{\beta \in H} M^\top(\beta, \alpha) \cdot \hat{f}_z(\beta) \right) \cdot r^\alpha \\
&= \sum_{\alpha \in H} \left(\hat{f}_M(\alpha) - \sum_{\beta \in H} M(\alpha, \beta) \cdot \hat{f}_z(\beta) \right) \cdot r^\alpha \qquad (4.3)
\end{aligned}$$

Completeness. Fix an instance $((\mathbb{F}, k, n, m, A, B, C, u), w) \in \mathcal{R}_{\mathsf{R1CS}}$. 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 4.3 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 $(\mathbb{F}, k, n, m, A, B, C, u) \notin L(\mathcal{R}_{\mathsf{R1CS}})$ and a prover $\tilde{\mathbf{P}}$. Let $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h} \in \mathbb{F}^{\leq |H|-1}[X]$ and $\hat{f}_w \in \mathbb{F}^{\leq |H|-k-1}[X]$ 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 Theorem 2.7.1, $\mathbf{V}$ with probability at least $1 - (|H| - 1)/|\mathbb{F}|$ over the choice of $a \leftarrow \mathbb{F}$:

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

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 $(\mathbb{F}, k, n, A, B, C, u) \notin L(\mathcal{R}_{\mathsf{R1CS}})$, 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 $1 - (|H| - 1)/|\mathbb{F}|$ over the choice of r, $\hat{g}(r) \neq 0$. If this is the case, by Equation 4.3 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 $1 - 2(|H| - 1)/|\mathbb{F}|$.

We conclude that $\mathbf{V}$ accepts with probability at most $3(|H| - 1)/|\mathcal{L}| = 3 \cdot (n - 1)/|\mathbb{F}|$.

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 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 $n - 1$.

□

4.7 Applications

In this section we describe further applications of our theorems:

- In Section 4.7.1 we plug in our proximity test for Reed–Solomon codewords into the generic compiler described in Section 4.4, giving a compiler from poly-IOPPs to IOPPs that can be used with any poly-IOPP.
- In Section 4.7.2 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 4.7.3 we give a high-soundness small-query proximity test for bivariate Reed–Muller codes.

4.7.1 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 4.7.1. *Let relation $\mathcal{R}$ 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 $\mathsf{q}_{\mathsf{poly},\ell}$ 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 $\mathcal{R}$ with the following parameters:

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

Proof. Let $\mathcal{L} \subseteq \mathbb{F}$ be a domain with $|\mathcal{L}| = (d_{\mathsf{max}}+1)/\rho$, and let $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d_{\mathsf{max}}]$ be a Reed–Solomon code whose rate is ρ. We apply Theorem 4.4.1 with the following ingredients:

- The poly-IOPP for $\mathcal{R}$ given in the theorem statement.
- The IOPP for $\mathcal{C}$ given by Theorem 4.5.1. Observe that $\sqrt{|\mathbb{F}|} \geq |G| \geq 2^8 \cdot |\mathcal{L}|$.
- The proximity generator given by Item 2 of Theorem 4.3.2 for $2\mathsf{q}_{\mathsf{poly},\ell}$ functions with proximity bound $\sqrt{\rho}$, seed length $2\mathsf{q}_{\mathsf{poly},\ell}\cdot\log|\mathbb{F}|$, error $O\left(\frac{d_{\mathsf{max}}^2\cdot(1/\rho)^4}{|\mathbb{F}|}\right)$ and computation time $O(\mathsf{q}_{\mathsf{poly},\ell})$.

- $\gamma := 1 - 2\sqrt{\rho}$.

The resulting IOPP for $\mathcal{R}$ 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

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

- ***Rounds.*** The number of rounds is $2\mathsf{k}_{\mathsf{poly}} + O(\log\log d_{\mathsf{max}}) + 1 = 2\mathsf{k}_{\mathsf{poly}} + O(\log\log d_{\mathsf{max}})$.
- ***Proof length.*** The proof length is

$$O(s_{\mathsf{poly}} \cdot |\mathcal{L}| + \mathsf{q}_{\mathsf{poly},\Pi} + |\mathcal{L}|/\rho) = O\left(\frac{d_{\mathsf{max}}}{\rho} \cdot \left(s_{\mathsf{poly}} + \frac{1}{\rho}\right) + \mathsf{q}_{\mathsf{poly},\Pi}\right).$$

- ***Oracle input queries.*** The verifier makes $\mathsf{q}_{\mathsf{poly},\mathbb{y}}$ queries to its input.
- ***Proof queries.*** The proof query complexity is $O(\mathsf{q}_{\mathsf{poly},\Pi} + \mathsf{q}_{\mathsf{poly},\ell} + \log\log d_{\mathsf{max}})$.
- ***Randomness.*** The randomness complexity is $\mathsf{r}_{\mathsf{poly}} + (\mathsf{k}_{\mathsf{poly}} + \mathsf{q}_{\mathsf{poly},\ell} + O(\log\log d_{\mathsf{max}})) \cdot \log|\mathbb{F}|$.
- ***Verifier running time.*** The verifier running time is $O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{q}_{\mathsf{poly},\ell} + \mathsf{k}_{\mathsf{poly}}) + \tilde{O}(\mathsf{q}_{\mathsf{poly},\Pi} + \sqrt{d_{\mathsf{max}}}) = O(\mathsf{vt}_{\mathsf{poly}}) + \tilde{O}(\mathsf{q}_{\mathsf{poly},\ell} + \sqrt{d_{\mathsf{max}}})$.

□

4.7.2 IOPPs for RS codes over every domain

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

Theorem 4.7.2. *Consider the following ingredients:*

- *A Reed–Solomon code* $\mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$.
- *An IOPP* $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *for* $\mathcal{C}_{\mathsf{prx}} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_{\mathsf{prx}}, d_{\mathsf{prx}}]$. *with rate* $\rho_{\mathsf{prx}} := (d_{\mathsf{prx}} + 1)/|\mathcal{L}_{\mathsf{prx}}|$.

- Gen *is proximity generator for* $\mathcal{C}_{\mathsf{prx}}$, 2 *functions, seed length* $\mathsf{w}_\star$, *proximity bound* $\psi_\star$, *and error* $\varepsilon_\star$.

If $d \leq d_{\mathsf{prx}}$, $0 < \gamma < 1 - \max\{\psi_\star, 2\rho_{\mathsf{prx}}\}$ *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}	$\max\{1-\delta, \beta_{\mathsf{prx}} + \mathsf{err}^\star + d_{\mathsf{prx}} \cdot \ell^2/\lvert\mathbb{F}\rvert\}$
Rounds	$\mathsf{k}_{\mathsf{prx}}$	$\mathsf{k}_{\mathsf{prx}} + 3$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	$\mathsf{l}_{\mathsf{prx}}$	$O(\lvert\mathcal{L}_{\mathsf{prx}}\rvert) + \mathsf{l}_{\mathsf{prx}}$
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	$\mathsf{r}_{\mathsf{prx}}$	$\mathsf{r}_{\mathsf{prx}} + \mathsf{w}_\star + 2\log\lvert\mathbb{F}\rvert$
Verifier running time	$\mathsf{vt}_{\mathsf{prx}}$	$\mathsf{vt}_{\mathsf{prx}} + \mathsf{t}_\star + O(1)$

Above, $\mathsf{err}^\star := \mathsf{err}^\star(\gamma)$ *and* $\beta_{\mathsf{prx}} := \beta_{\mathsf{prx}}(\gamma)$.

Proof. We first present a poly-IOPP for $\mathcal{C}$ and then apply Theorem 4.4.1 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 $f\colon \mathcal{L} \to \mathbb{F}$,

1. $\mathbf{P}_{\mathsf{poly}}$: 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 $f \in \mathcal{C} := \mathsf{RS}'[\mathbb{F}, \mathcal{L}, d]$ then the interpolation done by $\mathbf{P}_{\mathsf{poly}}$ will be successful, and, by definition, $f(a) = \hat{f}(a)$ for every $a \in \mathcal{L}$. Therefore, $\mathbf{V}_{\mathsf{poly}}$ 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 $\mathcal{L}$. 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 $\log\lvert\mathcal{L}\rvert$, input-query complexity 1, proof-query complexity 1, and verifier running time $O(1)$ field operations.

We now apply Theorem 4.4.1 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 $0 < \gamma < 1 - \max\{\mathsf{B}, 2\rho_{\mathsf{prx}}\}$, and, that $\mathcal{C}_{\mathsf{prx}}$ is (γ, ℓ)–list decodable. We can therefore apply Theorem 4.4.1. The parameters of the resulting IOPP follow. □

4.7.3 Testing bivariate RM codes with inverse polynomial error

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

Theorem 4.7.3. *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 $\mathcal{L}_X$ (as per Theorem 4.5.5). 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	$O(\mathrm{loglog}\, d_{\mathsf{max}})$
Proof length	$O\left(\frac{d_{\mathsf{max}} \cdot \lvert\mathcal{L}_X\rvert}{\rho_X}\right)$
Oracle input queries	1
Proof queries	$O(\mathrm{loglog}\, d_{\mathsf{max}})$
Randomness	$O(\mathrm{loglog}\, d_{\mathsf{max}} \cdot \log \lvert\mathbb{F}\rvert)$
Verifier running time	$\tilde{O}(d_Y + \sqrt{d_{\mathsf{max}}})$

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

Proof. We begin by instantiating a poly-IOPP for the code $\mathcal{C}$ using Theorem 4.5.6 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 $\mathcal{C}_X := \mathsf{RS}'[\mathbb{F}, \mathcal{L}_X, d_X]$ described in Item 1 of Theorem 4.3.4 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 $|\mathbb{F}| \geq |\mathcal{L}_X|^2$ and $|H| \geq d_Y$, and so the requirements for Theorem 4.5.6 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	$\lvert\mathcal{L}_X\rvert + 3$
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 $\mathsf{q}_{\mathsf{poly},\ell} := 5$.

We now compile the above poly-IOPP for $\mathcal{C}$ into an IOPP for $\mathcal{C}$ using Theorem 4.7.1, 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 4.7.1 have been met. The resulting IOPP has the following parameters:

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

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

  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 $10 + O(\mathrm{loglog}\, d_{\mathsf{max}}) = O(\mathrm{loglog}\, d_{\mathsf{max}})$.
- ***Proof length.*** The proof length is $O\left(\frac{d_{\mathsf{max}}}{\rho} \cdot \left(|\mathcal{L}_X| + \frac{1}{\rho}\right) + 5\right) = O\left(\frac{d_{\mathsf{max}} \cdot |\mathcal{L}_X|}{\rho_X}\right)$.
- ***Oracle input queries.*** The verifier makes 1 queries to its input.
- ***Proof queries.*** The proof query complexity is $O(5 + 5 + \mathrm{loglog}\, d_{\mathsf{max}}) = O(\mathrm{loglog}\, d_{\mathsf{max}})$.
- ***Randomness.*** The randomness complexity is $4 \log |\mathbb{F}| + (3 + 5 + O(\mathrm{loglog}\, d_{\mathsf{max}})) \cdot \log |\mathbb{F}| = O(\mathrm{loglog}\, d_{\mathsf{max}} \cdot \log |\mathbb{F}|)$.
- ***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 $\mathcal{R}$	$\rightarrow$ *IOPP for* $\mathcal{R}$
Proximity error	β_{poly}	$\max\left\{\beta_{\mathsf{poly}},\ \max\left\{2\rho^{1/2},\ \rho^{1/4}\right\} + O\left(\frac{d^2 \cdot (1/\rho)^4}{\lvert\mathbb{F}\rvert}\right)\right\}$
Rounds	$\mathsf{k}_{\mathsf{poly}}$	$2\mathsf{k}_{\mathsf{poly}} + O(\mathrm{loglog}\, d_{\mathsf{max}})$
Alphabet	$\mathbb{F}$	$\mathbb{F}$
Proof length	s_{poly} *(polynomials)*	$O\left(\frac{d_{\mathsf{max}}}{\rho} \cdot \left(s_{\mathsf{poly}} + \frac{1}{\rho}\right) + \mathsf{q}_{\mathsf{poly},\Pi}\right)$
Oracle input queries	$\mathsf{q}_{\mathsf{poly},y}$	$\mathsf{q}_{\mathsf{poly},y}$
Proof queries	$\mathsf{q}_{\mathsf{poly},\Pi}$	$O(\mathsf{q}_{\mathsf{poly},\Pi} + \mathsf{q}_{\mathsf{poly},\ell} + \mathrm{loglog}\, d_{\mathsf{max}})$
Randomness	$\mathsf{r}_{\mathsf{poly}}$	$\mathsf{r}_{\mathsf{poly}} + (\mathsf{k}_{\mathsf{poly}} + \mathsf{q}_{\mathsf{poly},\ell} + O(\mathrm{loglog}\, d_{\mathsf{max}})) \cdot \log \lvert\mathbb{F}\rvert$
Verifier running time	$\mathsf{vt}_{\mathsf{poly}}$	$O(\mathsf{vt}_{\mathsf{poly}}) + \tilde{O}(\mathsf{q}_{\mathsf{poly},\ell} + \sqrt{d_{\mathsf{max}}})$

□

Chapter 5

STIR: Reed–Solomon proximity testing with fewer queries

5.1 Introduction

The Reed–Solomon (RS) proximity testing problem considers the setting where a verifier has query access to a function $f : \mathcal{L} \to \mathbb{F}$ and the goal is to distinguish, by querying f at few locations, whether f is the evaluation of a degree d polynomial (i.e., is a codeword of $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$) or f is far in relative Hamming distance from all codewords in $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$. An untrusted prover may help the verifier, and different models consider different types of help. Here, we consider interactive oracle proofs of proximity (IOPPs), wherein the verifier interacts with the prover and has oracle access to the prover's messages.

Testing proximity to RS codes with few queries is a powerful capability. From a theoretical perspective, it is a key building block in celebrated PCP constructions [BS08; Din07; Mie09] and more. Moreover, in practice, it enables highly efficient constructions of succinct non-interactive arguments (SNARGs) [BBHR18; BGKS20; BCIKS20].

Especially noteworthy is FRI (and its variants), which is an IOPP for RS codes that enjoys practical efficiency [BBHR18; BGKS20; BCIKS20]. The most practically efficient version of FRI is given in [BCIKS20] and its implementation underlies numerous SNARG-based real-world systems, including [Pol; Ris; Stab; Staa; Zks; San; Mid; Nep; Ola; Her]. These systems offer state-of-the-art technology that protects billions of dollars' worth of transactions across various blockchains.

Query complexity. Small query complexity of an IOPP is crucial for achieving a small argument size when the IOPP is compiled into a SNARK. The compilation is typically performed via the BCS transformation [BCS16], in which each verifier query contributes additional size to the resulting argument string. In more detail, the BCS transformation can be viewed as two steps: (i) compile the IOPP into a succinct *interactive* argument by using Merkle commitments to the prover's messages and opening these commitments wherever the verifier wishes to query; and then (ii) apply

the Fiat–Shamir transformation to the succinct interactive argument to obtain a non-interactive argument.

Thus, each query of the IOPP verifier leads to the argument prover sending an additional opening of a Merkle commitment, which the argument verifier must subsequently verify. Consequently, with other factors being equal, reducing the query complexity of the IOPP verifier reduces the argument size and the argument verifier time for the corresponding non-interactive argument.

Round by round soundness. The BCS transformation requires a strong soundness property of the IOP (or IOPP) called *round-by-round soundness*.[1] Informally, this soundness notion requires that every round of the IOP individually has "small soundness error" (which is stronger than merely requiring than the entire IOP has small soundness error). Hence, to establish the compiled SNARG's security, one must establish the IOP's round-by-round soundness. The round-by-round soundness of FRI was only recently established [Sta21; BGKTRTZ23].

5.1.1 A new Reed–Solomon proximity test

We give a concretely efficient IOP of proximity for Reed–Solomon codes with small query complexity.

Theorem 9 (informal). *Let* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *be a "nice" Reed–Solomon code (*$\mathcal{L}$ *is a multiplicative coset of* $\mathbb{F}^*$ *whose size is a power of* 2 *and* d *is a power of* 2*) and* $\lambda \in \mathbb{N}$ *be a security parameter. If* $|\mathbb{F}| > \Omega\left(\frac{\lambda \cdot 2^\lambda \cdot d^2 \cdot |\mathcal{L}|^{3.5}}{\log 1/\rho}\right)$*, then* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *has an IOPP with round-by-round soundness error* $2^{-\lambda}$*, round complexity* $O(\log d)$*, proof length* $O(|\mathcal{L}|)$*, and query complexity* $O\left(\log d + \lambda \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$.

The formal version of the above theorem along with tighter bounds for the soundness analysis and field size appears in Section 5.4. We refer to the IOPP in Theorem 9 as *STIR*, standing for "Shift To Improve Rate". STIR is a recursive protocol that repeatedly reduces the degree by a constant factor k thus achieving $O(\log_k d)$ rounds. Crucially, this reduction also "improves the rate" in the sense that the rate shrinks (with each recursion), which improves soundness and reduces the query complexity. In particular, a round of STIR reduces testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $|\mathcal{L}'| = |\mathcal{L}|/2$. The reduction in the rate leads to a decrease in query complexity in the next recursive step. Consequently, the query complexity decreases with each iteration of the recursion.

Comparison to prior works. FRI [BBHR18] is a concretely efficient IOPP for RS codes. A variant called DEEP-FRI [BGKS20] achieves better soundness while essentially preserving other efficiency measures. Later, [BCIKS20] improved the analysis of the [BBHR18] protocol, showing that it achieves better parameters and, in fact,

[1] More precisely, the BCS transformation requires a notion called state-restoration soundness, which is implied by round-by-round soundness.

subsumes previous variants, including DEEP-FRI. This version, which we refer to as FRI, is widely used in practice.

FRI has a similar structure to STIR, also with $O(\log_k d)$ rounds, where the degree is reduced by a factor of k from one round to the next. Informally, the main difference between the two is that FRI does not enjoy a decrease in rate between rounds, which requires more queries to achieve the same level of security. For λ bits of security, FRI uses $O\left(\lambda \cdot \frac{\log d}{\log 1/\rho}\right)$ queries while STIR uses $O\left(\log d + \lambda \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$. Moreover, STIR is arguably "simpler" than FRI, as STIR can be analyzed iteration by iteration (as opposed to FRI that demands a "global" analysis due to its structure). This also facilitates a simpler proof of round-by-round soundness.

[ACY23] gives an IOPP for RS codes with inverse-polynomial soundness error, round complexity $O(\text{loglog}\, d)$, and query complexity $O(\text{loglog}\, d)$. Due to its small round complexity, this protocol has the potential to achieve smaller query complexity than STIR but, in order to be useful in practice, this would require addressing significant problems with regard to concrete efficiency. For instance, to achieve λ bits of security, the query complexity of (the amplified version of) [ACY23] is $O\left(\lambda^2 \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$ (moreover, the verifier does not have sublinear runtime). We leave open the question of whether the protocol in [ACY23] can be made concretely efficient.

Experimental results. We implement STIR in Rust using the `arkworks` [ark] ecosystem for developing zkSNARKs. For comparison, we also implement FRI with a similar level of optimizations (e.g., Merkle tree pruning, proof of work, and so on). Both FRI and STIR can be deployed with provable security parameters or improved parameters based on conjectures related to list-decoding of RS codes. Mirroring real-wold deployments of FRI, we compare STIR and FRI basing both on conjectured security parameters.

Our experiments show that STIR's improved IOPP query complexity leads to a significant reduction in both the argument size and the number of hashes computed by the argument verifier for the compiled SNARG. Depending on the parameters chosen, the improvement in argument size ranges from 1.25× to 2.46×, and the improvement in verifier hash complexity ranges from 1.55× to 2.67×.

Additionally, our experiments show that the running time of the prover is comparable to that of FRI. Typically, proximity tests are used in a batched setting, where the prover's running time is primarily constrained by computing the initial commitment (prior to running the proximity test). In this setting, replacing FRI with STIR leaves the prover time essentially unchanged.

We give concrete examples in Table 5.1. For a 192-bit prime field, degree $d = 2^{22}$, rate $\rho = 1/4$, and the goal of 128 bits of security, our experiments yield an argument of size 94 KiB for STIR versus 154 KiB for FRI. For degree 2^{28} with rate $\rho = 1/2$, we get an argument size of 189 KiB for STIR versus 430 KiB for FRI. In both examples, the STIR verifier performs roughly half the number of hash computations performed by the FRI verifier, with a similar prover running time. See Section 5.5 for more details

about the implementation and a detailed comparison with FRI, including argument sizes, prover times, verifier times, and number of hashes computed by the verifier.

	$d = 2^{22}, \rho = 1/4$		$d = 2^{28}, \rho = 1/2$	
	STIR	FRI	STIR	FRI
Argument size	94 KiB	154 KiB	189 KiB	430 KiB
Verifier time	2.1 ms	1.9 ms	4.3 ms	5.5 ms
Verifier hashes	1521	2821	3451	8479
Prover time	14 s	11 s	640 s	420 s

Table 5.1: Comparison of STIR and FRI.

Overall, since STIR and FRI solve the same problem (testing proximity to nice RS codes), STIR can be used as a drop-in replacement for FRI.

A polynomial IOP compiler. In practice IOPs are often constructed by combining two ingredients: a *polynomial IOP* for the language of interest (or other related IOP variants) and an RS proximity test (such as FRI or STIR or some other protocol).[2] Prior compilers [BCRSVW19; ACY23] were not analyzed for round-by-round soundness and, even for standard soundness, did not achieve high soundness guarantees. To address these limitations, we provide a new compiler that (is concretely efficient and) achieves high round-by-round knowledge soundness (knowledge soundness is a stronger soundness notion, where we require that if the verifier accepts with high enough probability, a witness for the language can be extracted efficiently given the protocol transcript).

Moreover, to illustrate the how to use our new compiler, we also give a polynomial IOP for the NP-complete language R1CS similar to the one given in [BCRSVW19] and analyze its round-by-round soundness error. Then, using our compiler with the polynomial IOP for R1CS, we compare the performance of STIR and FRI in this application using a Python script that computes the argument sizes. As in the experimental results, STIR outperforms FRI regarding argument size, yielding an improvement ranging from 1.29× to 2.25×. For example, for a 192-bit prime field, instance size $n = 2^{24}$, rate $\rho = 1/2$, and the goal of 128 bits of security, STIR has argument size 220 KiB compared to 422 KiB for FRI.

5.1.2 Additional result: batch degree correction

Degree correction is a problem that commonly arises in applications of RS proximity tests. The (batch) degree correction problem is as follows. Given functions $f_1, \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$ and degrees $d_1, \ldots, d_m, d^*$ with $d^* \geq \max_{i \in [m]} \{d_i\}$, define a function $f^* \colon \mathcal{L} \to \mathbb{F}$ (possibly using randomness or interaction) such that: (a) if $f_i \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$ for every $i \in [m]$ then $f^* \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$; (b) if any f_i is δ-far from

[2] A polynomial IOP is an IOP where both honest and malicious provers send bounded-degree polynomials as their oracle messages.

$\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$ then (with high probability) f^* is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$; and (c) query access to f^* can be efficiently simulated given query access to $f_1, \ldots, f_m$.[3] Below $\rho := d^*/|\mathcal{L}|$.

Prior work provides various solutions to this problem. [BCIKS20] gives a technique for the special case $d_1 = \cdots = d_m = d^*$ for $\delta \in (0, 1 - \sqrt{\rho})$; answering each query to f^* requires performing $O(m)$ operations. [BCRSVW19] shows (implicitly in their proof) a technique for the general case that works for $\delta \in (0, \frac{1-2\rho}{2})$ (where $\rho := d^*/|\mathcal{L}|$) and where answering each query requires performing $O(m \cdot \log d^*)$ operations. For the same technique, [ACY23] improve the bound to $\delta \in (0, \min\{1 - \sqrt{\rho}, 1 - 2 \cdot \rho\})$ (also implicitly in their proof).

We provide a concretely-efficient protocol for (batch) degree correction that further improves the bound on δ. We use this protocol in STIR and our efficient polynomial IOP compiler (yielding further concrete efficiency gains). Beyond these examples, our protocol can be used in other places where degree correction is needed (e.g., this would improve the compiler in [BCRSVW19]).

Theorem 10 (informal)**.** *There is a probabilistic transformation* Combine *such that for every functions $f_1, \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$, degrees $d_1, \ldots, d_m, d^*$ with $d^* \geq \max_{i \in [m]}\{d_i\}$, and distance $\delta \in (0, \min\{1 - \sqrt{\rho}, 1 - (1 + 1/d^*) \cdot \rho\})$, if*

$$\begin{aligned} \Pr\left[\Delta(\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]) \leq \delta\right] \\ > \mathsf{err}^\star\left(d^*, \rho, \delta, m \cdot (d+1) - \sum d_i\right), \end{aligned}$$

then each f_i is δ-close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$. Moreover, the functions have correlated agreement*: there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$ such that*

$$\forall\, i \in [\ell],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i],\ f_i(S) = u(S)\,.$$

Finally, any entry of the function $\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m))$ can be computed by reading a single entry from each of $f_1, \ldots, f_m$ and performing $O(m \cdot \log d^*)$ operations.

In the above theorem, $\mathsf{err}^\star$ is the error defined in [BCIKS20, Theorem 1.2] which satisfies

$$\mathsf{err}^\star(d, \rho, \delta, \ell) \leq \frac{(\ell - 1) \cdot d^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{1 - \sqrt{\rho} - \delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7}.$$

[3] Degree correction is useful even when $m = 1$, in which case one obtains a reduction from testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$ for $d^* \geq d$. This is useful when one only has a tester for the latter code; indeed, we rely on this case in STIR.

5.2 Techniques

We outline the main ideas behind our results. In Section 5.2.1, we present the basic properties of STIR and explain how these lead to improved query complexity. In Section 5.2.2, we describe and analyze a single iteration of STIR. In Section 5.2.3, we describe a technique to do batch degree correction for functions of different degrees.

5.2.1 Overview of STIR

We discuss the high-level structure of STIR. The goal is to test whether a function is close to the Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ where $\mathbb{F}$ is a finite field, $\mathcal{L}$ is a "smooth" subset of $\mathbb{F}$ of size n (i.e., a multiplicative coset of $\mathbb{F}^*$ whose size is a power of 2), and d is a power of 2. Throughout, $\rho := d/n$ is the *rate* of this Reed–Solomon code.

Outline of STIR. STIR is parameterized by a "folding parameter" $k \geq 4$ (a power of 2) and a query repetition parameter t. An iteration of STIR reduces testing proximity to the code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to a related code $\mathcal{C}' := \mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $|\mathcal{L}'| = n/2$, i.e., STIR reduces the degree by a factor of k while decreasing the size of the evaluation domain only by a factor of 2.

Testing proximity to $\mathcal{C}'$ is easier than testing proximity to $\mathcal{C}$ for two reasons: (i) $\mathcal{C}'$ is "smaller" than $\mathcal{C}$ in the sense that the degree is reduced from d to d/k and the evaluation domain size is reduced from n to $n/2$; (ii) the rate of the code is reduced from $\rho = \frac{d}{|\mathcal{L}|}$ to $\rho' = \frac{d/k}{|\mathcal{L}'|} = \frac{2}{k} \cdot \rho$. Intuitively, testing proximity to a code with a smaller rate is easier because the code has more "redundancy". Indeed, rate reduction in STIR is the key feature that facilitates smaller query complexity (the smaller the rate, the fewer queries are needed to achieve a target soundness error), as we discuss later in this section. Similar ideas of rate reduction have proven useful elsewhere to achieve efficient IOPPs [RR20; RR22].

The proof length of a single iteration is roughly $n/2$, and the verifier's query complexity is t (over an alphabet consisting of tuples of k field elements). A STIR iteration additionally amplifies distance: roughly, given a function that is δ-far from $\mathcal{C}$, except with probability $(1-\delta)^t$, the new function has distance $1 - \sqrt{\rho'}$ from $\mathcal{C}'$.

STIR consists of $M := O(\log_k d)$ iterations of this base protocol, reducing testing proximity to the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}', O(1)]$ where $|\mathcal{L}'| = n/2^M$. In iteration $i \in \{0, 1, \ldots, M-1\}$,[4] testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$ is reduced to testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}_{i+1}, d/k^{i+1}]$ where $|\mathcal{L}_{i+1}| = |\mathcal{L}_i|/2$ and the repetition parameter is t_i. As we see later, the improvement in rate in each round allows us to use a decreasing sequence of repetition parameters $t_0 \geq t_1 \geq \cdots \geq t_M$, starting with the given parameter $t_0 := t$. Once a constant degree is reached, proximity to this last Reed–Solomon code is tested using the standard test for constant-degree codes: the prover sends the entire constant-degree polynomial to the verifier, who compares the function being tested to this polynomial at t_M random locations.

[4] We index the repetitions starting from 0 rather than 1 for notational convenience.

STIR has query complexity $\sum_{i=1}^{M} t_i$ and proof length $\sum_{i=1}^{M} |\mathcal{L}_i|$ which is $O(n)$ since $|\mathcal{L}_i| = |\mathcal{L}|/2^i$. If the initial function has distance δ from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$, then the round-by-round[5] soundness error of the protocol is roughly $\varepsilon := \max\left\{(1-\delta)^{t_0}, \rho_1^{t_1/2}, \ldots, \rho_M^{t_M/2}\right\}$, where ρ_i is the rate of $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$.

Making fewer queries by improving rate. The improvement in rate and subsequent decrease in the required repetitions to achieve security is at the heart of how STIR achieves small query complexity. Given a desired security parameter λ, we set parameters such that the round-by-round soundness error is bounded by $2^{-\lambda}$. To this end, we set $t_0 := \frac{\lambda}{-\log(1-\delta)}$ and $t_i := \frac{\lambda}{-\log\sqrt{\rho_i}}$ (ignoring rounding issues) to get error $2^{-\lambda}$. Then, the query complexity in each iteration decreases: since $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$ has rate

$$\rho_i := \frac{d}{k^i} \cdot \frac{1}{|\mathcal{L}_i|} = \left(\frac{2}{k}\right)^i \cdot \frac{d}{n} = \left(\frac{2}{k}\right)^i \cdot \rho\,.$$

the query complexity at round i is

$$t_i := \frac{\lambda}{-\log\left((2/k)^{i/2} \cdot \sqrt{\rho}\right)} = \frac{2 \cdot \lambda}{i \cdot \log(k/2) - 2 \cdot \log\sqrt{\rho}}\,.$$

Thus, the verifier queries the input function at $t_0 = \frac{\lambda}{-\log(1-\delta)}$ locations (this is optimal for this soundness error) and the total proof query complexity is $\sum_{i=1}^{M} t_i = O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$.

Comparison with FRI. FRI [BBHR18] with folding parameter k also reduces testing proximity to a Reed–Solomon code to testing proximity to a Reed–Solomon code with smaller degree. FRI consists of multiple iterations where in iteration i the problem of testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$ is reduced to testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}_{i+1}, d/k^{i+1}]$ and a final test for constant-degree codes.

FRI differs from STIR in that $|\mathcal{L}_{i+1}| = |\mathcal{L}_i|/k$ (as opposed to $|\mathcal{L}_i|/2$), so that $|\mathcal{L}_i| = n/k^i$. As a result, the code associated to iteration i has rate $\rho_i := \frac{d}{k^i} \cdot \frac{k^i}{n} = \rho$. In other words, the rates in FRI remain fixed. Consequently, to achieve soundness $2^{-\lambda}$, the protocol must make at least $\frac{\lambda}{-\log\sqrt{\rho}}$ queries *in every round* except for the first. In fact, due to how FRI makes *correlated* queries to its iterations, all rounds *including the first* will have the same query complexity $t := \max\left\{\frac{\lambda}{-\log(1-\delta)}, \frac{\lambda}{-\log\sqrt{\rho}}\right\}$. As a result, the input query complexity of FRI is t and its proof query complexity is $\sum_{i=1}^{M} t = O_k\left(\lambda \cdot \left(\frac{\log d}{-\log(1-\delta)} + \frac{\log d}{-\log\sqrt{\rho}}\right)\right)$ which for reasonable settings of λ is a significantly larger dependence on d and $1/\rho$ when compared to STIR.

[5] Recall that round-by-round soundness is a strong soundness notion that turns out to be more important than standard soundness when compiling IOPs into SNARGs. See Section 2.1 for a precise definition.

Concrete parameters. The number of repetitions in both STIR and FRI is determined by the security analysis, which in turn relies on facts about the list-decoding of Reed–Solomon codes. As there are gaps in our understanding of Reed–Solomon codes, it is possible that the actual security of both protocols is higher, requiring fewer repetitions. Indeed, in real-world applications, the soundness of FRI is assumed higher based on a "List-Decoding Conjecture", which posits that the distance of the function following an iteration is (roughly) $(1-\rho')$-far from its corresponding code as opposed to $1-\sqrt{\rho'}$. As a result, the repetitions in a round of the protocol can be reduced by a factor of two to be $\frac{\lambda}{-\log \rho}$. Similarly, by adopting a comparable conjecture, we can decrease the number of repetitions of in STIR to $t_i := \frac{\lambda}{-\log \rho_i}$.

5.2.2 Anatomy of a STIR iteration

We describe a single iteration of STIR, reducing the goal of testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $\mathcal{L} := \langle \omega \rangle$ is generated by ω and $\mathcal{L}' := \omega \cdot \langle \omega^2 \rangle$. The detailed protocol allows for a more general setting of $\mathcal{L}$ and $\mathcal{L}'$. Before describing an iteration of STIR, we introduce the concepts of *folding* and *quotienting*.

Folding Reed–Solomon codewords. The *k-wise folding* of a function $f\colon \langle \omega \rangle \to \mathbb{F}$ at a point $r \in \mathbb{F}$ is a function $f_r := \mathsf{Fold}(f, r)\colon \langle \omega^k \rangle \to \mathbb{F}$. The function f_r at a point $x \in \langle \omega^k \rangle$ is defined as the output of $\hat{p}(r)$, where $\hat{p}$ is the unique polynomial of degree less than k such that $\hat{p}(y) = f(y)$ for every $y \in \langle \omega \rangle$ with $y^k = x$. This mapping has been used in prior low degree tests (e.g., [BBHR18; BGKS20; ACY23]), and has the following properties (see Section 5.3.4 for a proof of these properties):

1. If $f \in \mathsf{RS}[\mathbb{F}, \langle \omega \rangle, d]$, then, for every r, $f_r \in \mathsf{RS}[\mathbb{F}, \langle \omega^k \rangle, d/k]$; and
2. If f is δ-far from $\mathsf{RS}[\mathbb{F}, \langle \omega \rangle, d]$ for $\delta \in (0, 1-\sqrt{\rho})$, then with probability at least $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$ over the choice of r, f_r is δ-far from $\mathsf{RS}[\mathbb{F}, \langle \omega^k \rangle, d/k]$.

Furthermore, this mapping is local: given r and oracle access to f, $\mathsf{Fold}(f, r)$ can be computed at any point by reading a k tuple of field elements from f.

Univariate function quotienting. The *quotient* of a function $f\colon \langle \omega \rangle \to \mathbb{F}$ relative to $p\colon S \to \mathbb{F}$ with $S \subseteq \mathbb{F}$ is defined as:

$$\mathsf{Quotient}(f, S, p)(x) := \frac{f(x) - \hat{p}(x)}{\prod_{a \in S}(x - a)},$$

where $\hat{p}$ is the unique polynomial of degree less than $|S|$ such that $\hat{p}(a) = p(a)$ for every $a \in S$. Provided that $\langle \omega \rangle$ and S do not intersect, the quotient has the following properties (see Section 5.3.2 for a proof of these facts):

1. If $f \in \mathsf{RS}[\mathbb{F}, \langle \omega \rangle, d]$ is the evaluation on $\langle \omega \rangle$ of a polynomial of degree less than d that agrees with p on S, then $\mathsf{Quotient}(f, S, p) \in \mathsf{RS}[\mathbb{F}, \langle \omega \rangle, d - |S|]$.

2. If every polynomial $\hat{u}$ of degree less than d that is δ-close to f on $\langle\omega\rangle$ satisfies $\hat{u}(a) \neq p(a)$ for some $a \in S$, then $\mathsf{Quotient}(f, S, p)$ is δ-far from $\mathsf{RS}[\mathbb{F}, \langle\omega\rangle, d - |S|]$.

This mapping is local: given p and oracle access to f, $\mathsf{Quotient}(f, S, p)$ can be computed at any point of $\langle\omega\rangle$ with a single query to f.

The protocol. We describe an iteration of STIR, which reduces the goal of testing that f is close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing that a function f' is close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $\mathcal{L} := \langle\omega\rangle$ and $\mathcal{L}' := \omega \cdot \langle\omega^2\rangle$.[6]

1. *Sample folding randomness:* The verifier samples and sends $\alpha \leftarrow \mathbb{F}$.
2. *Send folded function:* The prover sends a function $g \colon \mathcal{L}' \to \mathbb{F}$. In the honest case, g is the evaluation of the polynomial $\hat{g}$ over $\mathcal{L}'$, where $\hat{g}$ is the extension of $\mathsf{Fold}(f, \alpha)$ to a polynomial of degree less than d/k.
3. *Out-of-domain sample:* The verifier samples and sends $r^{\text{out}} \leftarrow \mathbb{F} \setminus \mathcal{L}'$.
4. *Out-of-domain reply:* The prover sends a field element $\beta \in \mathbb{F}$. In the honest case, $\beta := \hat{g}(r^{\text{out}})$.
5. *Shift queries:* The verifier, for every $i \in [t]$, samples $z_i \leftarrow \langle\omega^k\rangle$ and obtains $y_i := f_\alpha(z_i)$ by querying the (virtual) oracle f_α, where $f_\alpha := \mathsf{Fold}(f, \alpha)$.

The next function f' is defined as $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ where $\mathcal{G} := \{r^{\text{out}}, z_1, \ldots, z_t\}$ and $p \colon \mathcal{G} \to \mathbb{F}$ is the function such that $p(r^{\text{out}}) = \beta$ and $p(z_i) = y_i$. Observe that the verifier has virtual oracle access to f' through its oracle access to g.

In Figure 5.1, we illustrate which functions are sent in the protocol and which are virtually derived with their corresponding evaluation domains.

The protocol has perfect completeness which directly follows from the honest prover's strategy described above and the properties of the fold and quotient operations. We discuss the complexity measures protocol of the protocol and its soundness error.

Analysis. The prover sends one oracle of length $|\omega \cdot \langle\omega^2\rangle| = |\langle\omega\rangle|/2$ plus an additional field element sent as a non-oracle message. The query complexity is t. Next, we discuss soundness.

Lemma 4. *If f is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ then f' is (approximately) $(1 - \sqrt{\rho'})$-far from the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$, except with probability $(1 - \delta)^t + \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$.*

Proof sketch.

1. With high probability, the function $f_\alpha := \mathsf{Fold}(f, \alpha)$ is δ-far from low degree. Specifically, by the properties of the folding function, $\Delta(f_\alpha, \mathsf{RS}[\mathbb{F}, \langle\omega^k\rangle, d/k]) \geq \delta$ with probability at least $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$.

[6] More precisely, the protocol as described reduces testing f to testing that f' is close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$ for a certain small set $\mathcal{G}$. We discuss the disparity between the degrees later on in this section.

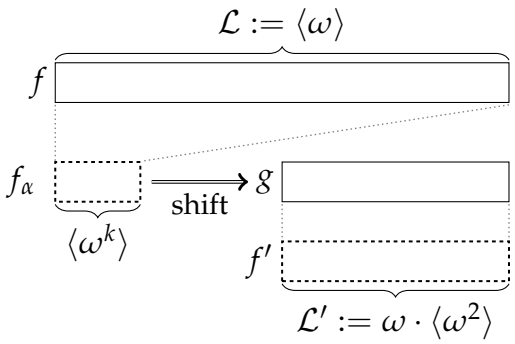


Figure 5.1: The basic structure of STIR: given α, the function f virtually defines $f_\alpha := \mathsf{Fold}(f, \alpha)$, which is (virtually) evaluated over $\langle \omega^k \rangle$. The prover then sends g evaluated over the larger domain $\mathcal{L}'$. Finally, the function $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ is virtually defined by quotienting g.

2. With high probability there is at most one codeword at distance $1 - \sqrt{\rho'}$ of g that evaluates to β at r^{out}. In more detail, with probability $1 - \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$ there exists at most one codeword $u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ at distance $\approx 1 - \sqrt{\rho'}$ from g with $\hat{u}(r^{\text{out}}) = \beta$, where $\hat{u}$ is the extension of u to a unique degree d/k polynomial.

   To see this, by the Johnson bound, the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ is (γ, ℓ)-list-decodable for $\gamma \approx 1 - \sqrt{\rho'}$ and $\ell = \text{poly}(|\mathcal{L}'|) = \text{poly}(|\mathcal{L}|)$, meaning that there are at most ℓ polynomials of degree less than d/k at distance γ to g. Each pair of such polynomials agree on less than d/k points, and so the total number of points in $\mathbb{F}$ for which there exist two distinct polynomials that are γ-close to g that agree on these points is bounded by $\binom{\ell}{2} \cdot d/k = O(\ell^2 \cdot d/k)$. One such point is sampled from $\mathbb{F}$ with probability at most $O(\ell^2 \cdot d/(k \cdot |\mathbb{F}|)) = \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$.

If Item 1 and Item 2 both hold, which happens with probability $1 - \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$, then f' is (approximately) $(1 - \sqrt{\rho'})$-far from the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$ with probability at least $1 - (1 - \delta)^t$. To see this, consider the following two cases.

- If there is no codeword u as in Item 2, then $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ is $(1 - \sqrt{\rho'})$-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$ since $p(r^{\text{out}}) = \beta$.

- If there exists a codeword u as in Item 2, then by Item 1 the polynomial $\hat{u}$ agrees with $\mathsf{Fold}(f, \alpha)$ on at most a $1 - \delta$ fraction of the domain. Thus, the probability that none of the t samples z_i hits such a location is at most $(1 - \delta)^t$. If a point z_i is chosen such that $f_{\alpha_i}(z_i) \neq \hat{u}(z_i)$, then there is no polynomial $(1 - \sqrt{\rho'})$-close to g that simultaneously agrees with f on z_i and is equal to β at r^{out}. As a result, $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ is $(1 - \sqrt{\rho'})$-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$.

□

In fact, a more thorough analysis reveals that the protocol has round-by-round soundness error roughly $\max\left\{\frac{\text{poly}(|\mathcal{L}|)}{|\mathbb{F}|}, (1-\delta)^t\right\}$. See Section 5.4.2 for a detailed analysis.

Degree correction. The protocol described above reduces testing that f is close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing that f' is close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$. STIR requires the degree to be a power of 2, and so in order to continue iterating to further reduce the degree, we modify the protocol to correct the degree up to d/k. The procedure to correct the degree is described in general terms in Section 5.2.3.

5.2.3 Efficient degree correction

We discuss efficient degree correction. We seek a transformation that, given a function $f\colon \mathcal{L} \to \mathbb{F}$, an initial degree d, and a target degree $d^* \geq d$, outputs a function f^* such that:

1. if $f \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ then $f^* \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$;
2. if f is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ then with high probability f^* is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$; and
3. query access to f^* can be simulated efficiently given query access to f.

In our applications we would like δ to be as large as possible, as a higher distance translates to smaller query complexity. The problem presented above is sufficient for STIR, but can be generalized for batch-degree correction for multiple functions with varying degrees, as presented in Section 5.1.2. We discuss this more general case, later on in this section. This more general case is used in our polynomial IOP to IOP compiler (see Section 5.6).

Prior solutions. Degree correction was tackled (implicitly) in [BCRSVW19] and in [ACY23]. Both use the same technique: sample a random field element $r \leftarrow \mathbb{F}$, and output $f^*(x) := f(x) + r \cdot x^e \cdot f(x)$, where $e := d^* - d$. Letting $\rho := d^*/|\mathcal{L}|$, [BCRSVW19] show that this works provided that $\delta < \frac{1-2\cdot\rho}{2}$, and [ACY23] improve the analysis to show that it works for $\delta < \min\{1 - \sqrt{\rho}, 1 - 2\cdot\rho\}$ (which in turn can be improved to $\delta < 1 - 2\cdot\rho$ assuming the List-Decoding Conjecture described in Section 5.2.1).

Our degree correction. We provide a different method that we prove works provided that f has distance $\delta < \min\{1 - \sqrt{\rho}, 1 - (1 + 1/d^*)\cdot\rho\}$ or, assuming the List-Decoding Conjecture, $\delta < 1 - (1 + 1/d^*)\cdot\rho$. Our method is as follows: sample a random field element $r \leftarrow \mathbb{F}$ and define $f^*(x) = \sum_{i=0}^{e} r^i \cdot f_i(x)$, where $f_i(x) := x^i \cdot f(x)$ and $e := d^* - d$.

Item 1 holds by construction. Next we sketch the proof that Item 2 holds provided that $\delta < \min\{1 - \sqrt{\rho}, 1 - (1 + 1/d^*)\cdot\rho\}$.

By a theorem of [BCIKS20], if with probability at least $\mathsf{err}^\star(d^*, \rho, \delta, e+1)$ (as defined in Theorem 10) the function f^* is δ-close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$ for $\delta < 1 - \sqrt{\rho}$, then the functions f_i have *δ-correlated agreement*: there exists a set S with $|S| \geq (1-\delta)\cdot|\mathcal{L}|$

such that for every f_i there exists a polynomial $\hat{f}_i$ of degree less than d^* such that $f_i(x) = \hat{f}_i(x)$ for every $x \in S$.

The following claim implies that $f_0 = f$ agrees with a polynomial of degree bounded by $d = d^* - e$ polynomial over S, leading to the conclusion that f is δ-close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$.

Claim 2. $\deg(\hat{f}_i) < d^* - e + i$ *for every* i.

Proof sketch. We prove the claim via (reverse) induction on i. The base case is immediate since $\deg(\hat{f}_e) < d^*$. Assuming that $\deg(\hat{f}_{i+1}) < d^* - e + i + 1$, we show that $\deg(\hat{f}_i) < d^* - e + i$. Consider the polynomial $\hat{p}(X) := X \cdot \hat{f}_i(X)$ and observe that, since $\deg(\hat{f}_i) < d^*$, it holds that $\deg(\hat{p}) < d^* + 1$. By correlated agreement on S, for every $x \in S$:

$$\hat{p}(x) = x \cdot \hat{f}_i(x) = x \cdot f_i(x) = x^{i+1} \cdot f(x) = f_{i+1}(x) = \hat{f}_{i+1}(x)\,.$$

We conclude that $\hat{p}$ and $\hat{f}_{i+1}$ agree on all points of S. Since $|S| \geq (1-\delta) \cdot |\mathcal{L}| > (1 + 1/d^*) \cdot \rho \cdot |\mathcal{L}| = d^* + 1$ and $\deg(\hat{p}), \deg(\hat{f}_{i+1}) < d^* + 1$, it follows that $\hat{p}$ and $\hat{f}_{i+1}$ are identical. In particular, $\deg(\hat{p}) = \deg(\hat{f}_{i+1}) < d^* - e + i + 1$. Recalling that $\hat{p}(X) := X \cdot \hat{f}_i(X)$, it follows that $\deg(\hat{f}_i) < d^* - e + i$. □

Efficient evaluation of f^*. At first glance, the technique described above does not allow for efficient local access to f^* as described in Item 3. Indeed, as the sum of $e+1$ different functions, it naively takes $O(e)$ time to compute f^* at a single point given access to f. While usable for small values of e, if $e = \Omega(d)$ this computation method is inefficient. However, we observe that $f^*(x)$ can be computed much faster since it can be viewed as a geometric sum:

$$f^*(x) = \sum_{i=0}^{e} r^i \cdot f_i(x) = \sum_{i=0}^{e} (r \cdot x)^i \cdot f(x) = \begin{cases} f(x) \cdot \left(\frac{1-(r\cdot x)^{e+1}}{1-r\cdot x}\right) & \text{if } r \cdot x \neq 1 \\ f(x) \cdot (e+1) & \text{if } r \cdot x = 1 \end{cases}.$$

The right-most expression can be computed in $O(\log e)$ operations using a single query to f and repeated squaring.

Combining functions of varying degrees. More generally, we have functions $f_1, \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$ and degrees $d_1, \ldots, d_m$ that we wish to batch-degree-correct into a single function f^*. We extend our method to this setting as follows: sample a random field element $r \leftarrow \mathbb{F}$ and define $e_i = d^* - d_i$ and:

$$f^*(x) = \sum_{i=0}^{e_1} r^i \cdot x^i \cdot f_1(x) + r^{1+e_1} \cdot \sum_{i=0}^{e_2} r^i \cdot x^i \cdot f_2(x) + \cdots + r^{m-1+\sum_{j=1}^{m-1} e_j} \cdot \sum_{i=0}^{e_m} r^i \cdot x^i \cdot f_m(x)\,.$$

We show, using similar techniques to those previously described, that if there is any f_i that is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$ then with high probability f^* is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]$ provided that $\delta < \min\{1 - \sqrt{\rho}, 1 - (1 + 1/d^*) \cdot \rho\}$. Moreover, by again utilizing geometric sums, local access for f^* can be simulated in time $O(\sum_i \log e_i) = O(m \cdot \log d^*)$ given local access to $f_1, \ldots, f_m$.

5.3 Tools for Reed–Solomon codes

We describe tools for Reed–Solomon codes that we use in this chapter.

- In Section 5.3.1 we describe a theorem of [BCIKS20] showing that taking a random linear combination is a good proximity generator for Reed–Solomon codes.
- In Section 5.3.2 we describe the quotient of a univariate function and show that that if a function is "quotiented by the wrong value", then the output is far from a Reed–Solomon codeword.
- In Section 5.3.3 we describe "out-of-domain sampling", a method to reduce the Reed–Solomon list-decoding size of a given function.
- In Section 5.3.4 we describe how to "fold" a univariate function and show that this preserves the function's distance to the Reed–Solomon code.
- In Section 5.3.5 we give a novel technique for combining functions of varying degrees (or correcting the degree of a single function) with nearly no loss in the range of parameters.

5.3.1 Random linear combination as a proximity generator

The theorem below states that if the random linear combination of several functions is low-degree with high probability then all of the functions are close to low-degree, with correlated agreement.

Theorem 5.3.1 ([BCIKS20]). *Let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ be a Reed–Solomon code with rate $\rho := d/|\mathcal{L}|$, and let $\mathsf{B}(\rho) := \sqrt{\rho}$. For every $\delta \in (0, 1 - \mathsf{B}(\rho))$ and functions $f_1 \ldots, f_\ell \colon \mathcal{L} \to \mathbb{F}$, if*

$$\Pr_{r \leftarrow \mathbb{F}} \left[\Delta \left(\sum_{j=1}^{\ell} r^{j-1} \cdot f_j,\ \mathsf{RS}[\mathbb{F}, \mathcal{L}, d] \right) \leq \delta \right] > \mathsf{err}^\star(d, \rho, \delta, \ell)\,,$$

then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$, and

$$\forall\, i \in [\ell],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d],\ f_i(S) = u(S)\,.$$

Above, $\mathsf{err}^\star(d, \rho, \delta, \ell)$ is defined as follows:

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

$$\mathsf{err}^\star(d, \rho, \delta, \ell) := \frac{(\ell - 1) \cdot d}{\rho \cdot |\mathbb{F}|}\,.$$

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

$$\mathsf{err}^\star(d, \rho, \delta, \ell) := \frac{(\ell - 1) \cdot d^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{1 - \sqrt{\rho} - \delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7}\,.$$

5.3.2 Univariate function quotienting

We define the *quotient* of a univariate function.

Definition 5.3.2. *Let $f\colon \mathcal{L} \to \mathbb{F}$ be a function, $S \subseteq \mathbb{F}$ be a set, and $\mathsf{Ans}, \mathsf{Fill}\colon S \to \mathbb{F}$ be functions. Let $\hat{\mathsf{Ans}} \in \mathbb{F}^{<|S|}[X]$ be the (unique) polynomial with $\hat{\mathsf{Ans}}(x) = \mathsf{Ans}(x)$ for every $x \in S$, and let $\hat{V}_S \in \mathbb{F}^{<|S|+1}[X]$ be the unique non-zero polynomial with $\hat{V}_S(x) = 0$ for every $x \in S$.*

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

$$\forall\, x \in \mathcal{L}\,,\ \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(x) := \begin{cases} \mathsf{Fill}(x) & x \in S \\ \frac{f(x) - \hat{\mathsf{Ans}}(x)}{\hat{V}_S(x)} & \text{otherwise} \end{cases}.$$

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

Definition 5.3.3. *Let $\hat{f} \in \mathbb{F}^{<d}[X]$ be a polynomial and $S \subseteq \mathbb{F}$ be a set, and let $\hat{V}_S \in \mathbb{F}^{<|S|+1}[X]$ be the unique non-zero polynomial with $\hat{V}_S(x) = 0$ for every $x \in S$. The* **polynomial quotient** *$\mathsf{PolyQuotient}(\hat{f}, S) \in \mathbb{F}^{<d-|S|}[X]$ is defined as follows:*

$$\mathsf{PolyQuotient}(\hat{f}, S)(X) := \frac{\hat{f}(X) - \hat{\mathsf{Ans}}(X)}{\hat{V}_S(X)}\,,$$

where $\hat{\mathsf{Ans}} \in \mathbb{F}^{<|S|}[X]$ is the unique (nonzero) polynomial with $\hat{\mathsf{Ans}}(x) = \hat{f}(x)$ for every $x \in S$.

The following lemma, implicit in prior works (e.g., [BGKS20; ACY23]), shows that if a function is "quotiented by the wrong value", then its quotient is far from low-degree.

Lemma 5.3.4. *Let $f\colon \mathcal{L} \to \mathbb{F}$ be a function, $d \in \mathbb{N}$ be a degree parameter, $\delta \in (0,1)$ be a distance parameter, $S \subseteq \mathbb{F}$ be a set with $|S| < d$, and $\mathsf{Ans}, \mathsf{Fill}\colon S \to \mathbb{F}$ be functions. Suppose that for every $u \in \Lambda(f, d, \delta)$ there exists $x \in S$ with $\hat{u}(x) \neq \mathsf{Ans}(x)$. Then*

$$\Delta(\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d - |S|]) + |T|/|\mathcal{L}| > \delta\,,$$

where $T := \{x \in \mathcal{L} \cap S : \hat{\mathsf{Ans}}(x) \neq f(x)\}$.

Proof. Let $g := \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})$ and suppose towards contradiction that there exists a polynomial $\hat{g} \in \mathbb{F}^{<d-|S|}[X]$ that agrees with g on at least a $(1 - \delta + |T|/|\mathcal{L}|)$-fraction of the locations of $\mathcal{L}$. Consider the "unquotiented" polynomial $\hat{w}(X) = \hat{V}_S(X) \cdot \hat{g}(X) + \hat{\mathsf{Ans}}(X)$ where $\hat{\mathsf{Ans}}$ and $\hat{V}_S$ are defined as in Theorem 5.3.2. Observe that $\deg(\hat{w}) < d$ and that for every $x \in \mathcal{L} \setminus T$ where $\hat{g}(x) = g(x)$, we have

$$\hat{w}(x) = \hat{V}_S(x) \cdot \hat{g}(x) + \hat{\mathsf{Ans}}(x) = \hat{V}_S(x) \cdot g(x) + \hat{\mathsf{Ans}}(x) = f(x)\,.$$

The last equality follows by Theorem 5.3.2 since:

- if $x \in S \setminus T$ then $\hat{w}(x) = \hat{\mathsf{Ans}}(x) = f(x)$;
- if $x \notin S$ then $g(x) = \frac{f(x) - \hat{\mathsf{Ans}}}{\hat{V}_S(x)}$ so that $\hat{V}_S(x) \cdot g(x) + \hat{\mathsf{Ans}}(x) = f(x)$.

The number of points in $\mathcal{L} \setminus T$ with $g(x) = \hat{g}(x)$ is at least $(1 - \delta + |T|/|\mathcal{L}|) \cdot |\mathcal{L}| - |T| = (1 - \delta) \cdot |\mathcal{L}|$, so we deduce that $\hat{w}$ on $\mathcal{L}$ is δ-close to f. Moreover, for every $x \in S$ it holds that $\hat{w}(x) = \hat{\mathsf{Ans}}(x) = \mathsf{Ans}(x)$. This is a contradiction to the assumption in the lemma statement. □

5.3.3 Out of domain sampling

The following lemma shows that the probability that there exist two distinct codewords in the list-decoding set of a function that both agree on a random point is small.

Lemma 5.3.5. *Let $f \colon \mathcal{L} \to \mathbb{F}$ be a function, $d \in \mathbb{N}$ be a degree parameter, $s \in \mathbb{N}$ be a repetition parameter, and $\delta \in [0,1]$ be a distance parameter. If $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is (δ, ℓ)-list decodable then*

$$\Pr_{r_1,\ldots,r_s \leftarrow \mathbb{F}\setminus\mathcal{L}} \left[\exists \text{ distinct } u, u' \in \Lambda(f,d,\delta) : \forall\, i \in [s],\ \hat{u}(r_i) = \hat{u}'(r_i)\right] \leq \binom{\ell}{2} \cdot \left(\frac{d-1}{|\mathbb{F}| - |\mathcal{L}|}\right)^s \leq \frac{\ell^2}{2} \cdot \left(\frac{d}{|\mathbb{F}| - |\mathcal{L}|}\right)^s.$$

Proof. Fix two distinct codewords $u, u' \in \Lambda(f,d,\delta)$. Since $\hat{u}$ and $\hat{u}'$ are distinct and have degree less than d, $\Pr_{r \leftarrow \mathbb{F}\setminus\mathcal{L}}[\hat{u}(r) = \hat{u}'(r)] \leq \frac{d-1}{|\mathbb{F}|-|\mathcal{L}|}$, so the probability that the polynomials agree on points $r_1, \ldots, r_s$ is at most $\left(\frac{d-1}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$. Since the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is (δ, ℓ)-list decodable, there are at most $\binom{\ell}{2}$ pairs of distinct codewords u, u' at distance at most δ from f. By the union bound, the probability that, over a random choice of $r \in \mathbb{F}$, there exist distinct codewords u, u' at distance at most δ from f such that $\hat{u}(r) = \hat{u}'(r)$ is at most $\binom{\ell}{2} \cdot \left(\frac{d-1}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$. □

5.3.4 Folding univariate functions

STIR relies on k-wise "folding" of functions and polynomials. As shown below, folding a function preserves its proximity from the Reed–Solomon code with high probability. While described in slightly different form, this is identical to folding in prior works (e.g., [BBHR18; BGKS20; ACY23]).

The folding operator is based on the following fact, decomposing univariate polynomials into bivariate polynomials.

Fact 5.3.6 ([BS08])**.** *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* $\deg_X(\hat{Q}) = \left\lfloor \deg(\hat{f})/\deg(\hat{q}) \right\rfloor$ *and* $\deg_Y(\hat{Q}) < \deg(\hat{q})$ *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* $\deg(\hat{f}) < t \cdot \deg(\hat{q})$ *then* $\deg_X(\hat{Q}) < t$.
- *For every* $\hat{Q} \in \mathbb{F}[X,Y]$ *with* $\deg_X(\hat{Q}) < t$ *and* $\deg_Y(\hat{Q}) < \deg(\hat{q})$, *the polynomial* $\hat{f}(Z) := \hat{Q}(\hat{q}(Z), Z)$ *has degree* $\deg(\hat{f}) < t \cdot \deg(\hat{q})$.

We define the folding of a polynomial and then the folding of a function.

Definition 5.3.7. *Given a polynomial* $\hat{f} \in \mathbb{F}^{<d}[X]$, *a folding parameter* $k \in \mathbb{N}$, *and* $r \in \mathbb{F}$, *we define a polynomial* $\mathsf{PolyFold}(\hat{f}, k, r) \in \mathbb{F}^{<d/k}[X]$ *as follows. Let* $\hat{Q} \in \mathbb{F}[X,Y]$ *be the bivariate polynomial derived from* $\hat{f}$ *using Theorem 5.3.6 with* $\hat{q}(X) := X^k$. *Then* $\mathsf{PolyFold}(\hat{f}, k, r)(X) := \hat{Q}(X, r)$.

Definition 5.3.8. *Let* $f \colon \mathcal{L} \to \mathbb{F}$ *be a function,* $k \in \mathbb{N}$ *a folding parameter, and* $\alpha \in \mathbb{F}$. *For every* $x \in \mathcal{L}^k$, *let* $\hat{p}_x \in \mathbb{F}^{<k}[X]$ *be the polynomial where* $\hat{p}_x(y) = f(y)$ *for every* $y \in \mathcal{L}$ *such that* $y^k = x$. *We define* $\mathsf{Fold}(f, k, \alpha) \colon \mathcal{L}^k \to \mathbb{F}$ *as follows:*

$$\mathsf{Fold}(f, k, \alpha)(x) := \hat{p}_x(\alpha)\,.$$

In order to compute $\mathsf{Fold}(f, k, \alpha)(x)$ *it suffices to interpolate the* k *values* $\{f(y) : y \in \mathcal{L} \text{ s.t. } y^k = x\}$ *into the polynomial* $\hat{p}_x$ *and evaluate this polynomial at* α.

The following lemma shows that the distance of a function is preserved under folding. If f has distance δ to a given Reed–Solomon code then, with high probability over the choice of folding randomness, its folding also has distance δ to the "k-wise folded" Reed–Solomon code.

Lemma 5.3.9. *For every function* $f \colon \mathcal{L} \to \mathbb{F}$, *degree parameter* $d \in \mathbb{N}$, *folding parameter* $k \in \mathbb{N}$, *and distance parameter* $\delta \in (0, \min\{\Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]), 1 - \mathsf{B}(\rho)\})$, *letting* $\rho := d/|\mathcal{L}|$,

$$\Pr_{\alpha \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, \alpha), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \le \delta\right] \le \mathsf{err}^\star(d/k, \rho, \delta, k)\,.$$

Above, B *and* $\mathsf{err}^\star$ *are the proximity bound and error (respectively) described in Section 5.3.1.*

Proof. Suppose towards contradiction that

$$\Pr_{\alpha \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, \alpha), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \le \delta\right] > \mathsf{err}^\star(d/k, \rho, \delta, k)\,.$$

Letting $\hat{p}_x$ be defined from f as in Theorem 5.3.8, define $c_0, \ldots, c_{k-1}$ where $c_j \colon \mathcal{L}^k \to \mathbb{F}$ is the function where $c_j(x)$ is the j-th coefficient of $\hat{p}_x$ (i.e., so that $\hat{p}_x(X) \equiv \sum_{j=0}^{k-1} c_j(x) \cdot X^j$ for every $x \in \mathcal{L}^k$). Observe that

$$\mathsf{Fold}(f, k, \alpha)(x) = \hat{p}_x(\alpha) = \sum_{j=0}^{k-1} c_j(x) \cdot \alpha^j\,.$$

Therefore, we get that

$$\begin{aligned}
&\Pr_{\alpha \leftarrow \mathbb{F}}\left[\Delta\left(\sum_{j=0}^{k-1} c_j \cdot (\alpha)^j, \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]\right) \leq \delta\right] \\
&= \Pr_{\alpha \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, \alpha), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \leq \delta\right] \\
&> \mathsf{err}^*(d/k, \rho, \delta, k)\,.
\end{aligned}$$

By Theorem 5.3.1, there exists a set $S \subseteq \mathcal{L}^k$ with $|S| \geq (1-\delta) \cdot |\mathcal{L}^k|$ such that for every $j \in \{0, \ldots, k-1\}$ there exists a codeword $u_j \in \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]$ such that c_j and u_j agree on S.

Let $S' \subseteq S$ be a set with $|S'| = \min\{|S|, d/k\}$ and, for every $x \in S'$, let $I_{x,S'} \in \mathbb{F}^{<d/k}[X]$ be the indicator polynomial where $I_{x,S'}(x) = 1$ and $I_{x,S'}(y) = 0$ for every $y \in S' \setminus \{x\}$. Consider the following bivariate polynomial

$$\hat{Q}(X, Y) := \sum_{x \in S'} I_{x,S'}(X) \cdot \hat{p}_x(Y)\,.$$

The degrees of $\hat{Q}$ are $\deg_X(\hat{Q}) < d/k$ and $\deg_Y(\hat{Q}) < k$.

For every $\alpha \in \mathbb{F}$ and $x \in S'$, $\hat{Q}(x, \alpha) = \hat{p}_x(\alpha) = \sum_{j=0}^{k-1} c_j(x) \cdot \alpha^j = \sum_{j=0}^{k-1} \hat{u}_j(x) \cdot \alpha^j$. If $|S'| \geq d/k$ then, since the degree of $\hat{u}_j$ is d/k, it holds that $\hat{Q}(X, \alpha) \equiv \sum_{j=0}^{k-1} \hat{u}_j(X) \cdot \alpha^j$. Observing that $\hat{p}_x(\alpha) = \sum_{j=0}^{k-1} c_j(x) \cdot \alpha^j = \sum_{j=0}^{k-1} \hat{u}_j(x) \cdot \alpha^j$ also for $x \in S \setminus S'$, we deduce that $\hat{Q}(x, Y) \equiv \hat{p}_x(Y)$ for every $x \in S$. If $|S| < d/k$ then $S = S'$, and so this holds trivially.

Observe that the polynomial $\hat{f}(X) := \hat{Q}(X^k, X)$ has degree d. Moreover, by construction for every x with $x^k \in S$:

$$\hat{f}(x) = \hat{Q}(x^k, x) = \hat{p}_{x^k}(x) = f(x)\,.$$

Thus, there are at least $k \cdot |S| \geq k \cdot (1-\delta) \cdot |\mathcal{L}^k| = (1-\delta) \cdot |\mathcal{L}|$ points where $\hat{f}$ and f agree. This contradicts the fact that $\delta < \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d])$. □

5.3.5 Combining functions of varying degrees

We show a new method for combining functions of varying degrees with minimal proximity requirements using geometric sums. We begin by recalling a fact about geometric sums.

Fact 5.3.10. *Let $\mathbb{F}$ be a field, $r \in \mathbb{F}$ be a field element, and $a \in \mathbb{N}$ be a natural number. Then*

$$\sum_{i=0}^{a} r^i = \begin{cases} \left(\frac{1-r^{a+1}}{1-r}\right) & r \neq 1 \\ a+1 & r = 1 \end{cases}.$$

Definition 5.3.11. *Given target degree $d \in \mathbb{N}$, shifting parameter r, functions $f_1, \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$, and degrees $0 \leq d_1, \ldots, d_m \leq d^*$, we define* $\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)) \colon \mathcal{L} \to \mathbb{F}$ *as follows:*

$$\begin{aligned}\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m))(x) &:= \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{\ell=0}^{d^*-d_i} (r \cdot x)^\ell \right) \\ &= \begin{cases} \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\frac{1-(xr)^{d^*-d_i+1}}{1-xr} \right) & x \cdot r \neq 1 \\ \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot (d^* - d_i + 1) & x \cdot r = 1 \end{cases}.\end{aligned}$$

Above, $r_1 := 1$ and $r_i := r^{i-1+\sum_{j<i}(d^*-d_j)}$ for $i > 1$.

In cases when we only want to degree correct, but have no need for combining multiple functions we use the following explicit degree correction notation.

Definition 5.3.12. *Given target degree $d \in \mathbb{N}$, shifting parameter r, function $f \colon \mathcal{L} \to \mathbb{F}$, and degree $0 \leq d \leq d^*$, we define* $\mathsf{DegCor}(d^*, r, f, d) \colon \mathcal{L} \to \mathbb{F}$ *as follows:*

$$\mathsf{DegCor}(d^*, r, f, d)(x) := f(x) \cdot \left(\sum_{\ell=0}^{d^*-d} (r \cdot x)^\ell \right) = \begin{cases} f(x) \cdot \left(\frac{1-(xr)^{d^*-d+1}}{1-xr} \right) & x \cdot r \neq 1 \\ f(x) \cdot (d^* - d + 1) & x \cdot r = 1 \end{cases}.$$

(Observe that $\mathsf{DegCor}(d^*, r, f, d) \equiv \mathsf{Combine}(d^*, r, (f, d))$*.)*

We show that combining multiple polynomials of varying degrees can be done as long as the proximity error is bounded by $\min\{1 - \mathsf{B}(\rho), 1 - \rho - 1/|\mathcal{L}|\}$.

Lemma 5.3.13. *Let $d^* \in \mathbb{N}$ be a target degree, $f_1, \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$ be functions, $0 \leq d_1, \ldots, d_m \leq d^*$ be degrees, and $\delta \in (0, \min\{1 - \mathsf{B}(\rho), 1 - \rho - 1/|\mathcal{L}|\})$ be a distance parameter, where $\rho := d^*/|\mathcal{L}|$. If*

$$\begin{aligned}&\Pr_{r \leftarrow \mathbb{F}} \left[\Delta(\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]) \leq \delta \right] \\ &\quad > \mathsf{err}^\star \left(d^*, \rho, \delta, m \cdot (d^* + 1) - \sum_{i=1}^{m} d_i \right),\end{aligned}$$

then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$, and

$$\forall\, i \in [\ell],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i],\ f_i(S) = u(S)\,.$$

Note that this implies that $\Delta(f_i, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]) \leq \delta$ for every i. Above, B and $\mathsf{err}^\star$ are the proximity bound and error (respectively) described in Section 5.3.1.

Proof. By Theorem 5.3.11, for every r,

$$\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)) = \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{\ell=0}^{d^*-d_i} (r \cdot x)^\ell \right).$$

Then, since $r_1 = 1$ and $r_i := r^{i-1+\sum_{j<i}(d^*-d_j)}$:

$$\begin{aligned}
&\Pr_{r\leftarrow\mathbb{F}}\left[\Delta\left(\sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{\ell=0}^{d^*-d_i} (r\cdot x)^\ell\right), \mathsf{RS}[\mathbb{F},\mathcal{L},d^*]\right) \le \delta\right] \\
&= \Pr_{r\leftarrow\mathbb{F}}\left[\Delta(\mathsf{Combine}(d^*, r, (f_1,d_1),\ldots,(f_m,d_m)), \mathsf{RS}[\mathbb{F},\mathcal{L},d^*]) \le \delta\right] \\
&> \mathsf{err}^\star\left(d^*, \rho, \delta, m\cdot(d^*+1) - \sum_{i=1}^{m} d_i\right).
\end{aligned}$$

By Theorem 5.3.1, there exists a set $S \subseteq \mathcal{L}$ with $|S| > (1-\delta)\cdot|\mathcal{L}|$ such that for every $i \in [m]$ and $j \in \{0,\ldots,d^*-d_i\}$ there exists a polynomial $\hat{p}_{i,j} \in \mathbb{F}^{<d^*}[X]$ such that $\hat{p}_{i,j}(x) = x^j \cdot f_i(x)$ for every $x \in S$.

Fix $i \in [m]$. We inductively show that $\deg(\hat{p}_{i,j}) < d_i + j$. This proves the lemma since it implies that there is a polynomial $\hat{p}_{i,0} \in \mathbb{F}^{<d_i}[X]$ that agrees with $f_i(x)$ on all of the points in S, and this was true for any $i \in [m]$.

As the base case, it is immediate that $\deg(\hat{p}_{i,d^*-d_i}) < d^* = d_i + d^* - d_i$. For $0 \le j < d^* - d_i$ suppose that $\deg(\hat{p}_{i,j+1}) < d_i + j + 1$. We show that $\deg(\hat{p}_{i,j}) < d_i + j$. Consider the polynomial $\hat{q}(X) := X \cdot \hat{p}_{i,j}(X)$. Since $\deg(\hat{p}_{i,j}) < d$, it follows that $\deg(\hat{q}) < d^* + 1$. Observe that for every $x \in S$,

$$\hat{q}(x) = x \cdot \hat{p}_{i,j}(x) = x^{j+1} \cdot f_i(x) = \hat{p}_{i,j+1}(x).$$

The polynomials $\hat{q}$ and $\hat{p}_{i,j+1}$ have degree less than d^*+1, and agree on $|S| \ge (1-\delta)\cdot|\mathcal{L}| > (\rho + 1/|\mathcal{L}|)\cdot|\mathcal{L}| = d^*+1$ points. They are therefore identical and, in particular, $\deg(\hat{q}) = \deg(\hat{p}_{i,j+1}) < d^* + j + 1$. Recalling that $\hat{q}(X) := X \cdot \hat{p}_{i,j}(X)$ we conclude that $\deg(\hat{p}_{i,j}) < d_i + j$. □

5.4 STIR

We describe STIR, an interactive oracle proof of proximity for nice Reed–Solomon codes.

- In Section 5.4.1 we describe the construction and analyze its complexity parameters.
- In Section 5.4.2 we prove round-by-round soundness of STIR.
- In Section 5.4.3 we give recommended settings of parameters for STIR, including a numeric example.

Theorem 5.4.1. *Consider the following ingredients:*
- *A security parameter* $\lambda \in \mathbb{N}$.
- *A Reed–Solomon code* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *with rate* $\rho := d/|\mathcal{L}|$ *where* d *is a power of* 2*, and* $\mathcal{L}$ *is a smooth domain.*
- *A proximity parameter* $\delta \in (0, 1 - 1.05 \cdot \sqrt{\rho})$.
- *A folding parameter* $k \in \mathbb{N}$ *that is a power of* 2 *with* $k \geq 4$.

If $|\mathbb{F}| = \Omega\left(\frac{\lambda \cdot 2^\lambda \cdot d^2 \cdot |\mathcal{L}|^{3.5}}{\log(1/\rho)}\right)$*, there is a public-coin IOPP for* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *with the following parameters:*

- Round-by-round soundness error: $2^{-\lambda}$.
- Round complexity: $M := O(\log_k d)$.
- Proof length: $|\mathcal{L}| + O_k(\log d)$.
- Query complexity to the input: $\frac{\lambda}{-\log(1-\delta)}$.
- Query complexity to the proof strings: $O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$.

5.4.1 Construction

We describe STIR and analyze its complexity parameters.

Construction 5.4.2. Consider the following ingredients:

- a field $\mathbb{F}$;
- an iteration count $M \in \mathbb{N}$;
- an initial degree parameter $d \in \mathbb{N}$ that is a power of 2;
- folding parameters $k_0, \ldots, k_M \in \mathbb{N}$ that are powers of two, with $d \geq \prod_i k_i$;
- evaluation domains $\mathcal{L}_0, \ldots, \mathcal{L}_M \subseteq \mathbb{F}$ where $\mathcal{L}_i$ is a smooth coset of $\mathbb{F}^*$ with order $|\mathcal{L}_i| > d/\prod_{j<i} k_j$;[7]
- repetition parameters $t_0, \ldots, t_M \in \mathbb{N}$ where $t_i + 1 \leq d/\prod_{j \leq i} k_j$ for every $i \in \{0, \ldots, M-1\}$;
- out-of-domain repetition parameter $s \in \mathbb{N}$.

[7] If, additionally, $\mathcal{L}_i \cap \mathcal{L}_{i+1} = \emptyset$ for every i, then the protocol can be made more efficient. See Theorem 5.4.3.

For every $i \in \{0, \ldots, M\}$, set $d_i := \frac{d}{\prod_{j<i} k_j}$. The protocol proceeds as follows.

- **Initial function:** Let $f_0 \colon \mathcal{L}_0 \to \mathbb{F}$ be an oracle function. In the honest case, $f_0 \in \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0]$ and the prover has access to the polynomial $\hat{f}_0 \in \mathbb{F}^{<d_0}[X]$ whose restriction to $\mathcal{L}_0$ is f_0.
- **Initial folding:** The verifier sends $\alpha_0 \leftarrow \mathbb{F}$.
- **Interaction phase loop:** For $i = 1, \ldots, M$:
  1. **Send folded function:** The prover sends a function $g_i \colon \mathcal{L}_i \to \mathbb{F}$. In the honest case, g_i is the evaluation of the polynomial $\hat{g}_i := \mathsf{PolyFold}(\hat{f}_{i-1}, k_{i-1}, \alpha_{i-1})$ over $\mathcal{L}_i$.
  2. **Out-of-domain samples:** The verifier sends $r^{\text{out}}_{i,1}, \ldots, r^{\text{out}}_{i,s} \leftarrow \mathbb{F} \setminus \mathcal{L}_i$.
  3. **Out-of-domain reply:** The prover sends field elements $\beta_{i,1}, \ldots, \beta_{i,s} \in \mathbb{F}$. In the honest case, $\beta_{i,j} := \hat{g}_i(r^{\text{out}}_{i,j})$.
  4. **STIR message:** The verifier sends $\alpha_i, \gamma_i \leftarrow \mathbb{F}$ and $z_{i,1}, \ldots, z_{i,t_{i-1}} \leftarrow \mathcal{L}_{i-1}^{k_{i-1}}$.
  5. **Define next polynomial and send hole fills:** The prover sends oracle message $\mathsf{Fill}_i \colon \{z_{i,1}, \ldots, z_{i,t_{i-1}}\} \cap \mathcal{L}_i \to \mathbb{F}$. In the honest case, the prover defines $\mathcal{G}_i := \{r^{\text{out}}_{i,1}, \ldots, r^{\text{out}}_{i,s}, z_{i,1}, \ldots, z_{i,t_{i-1}}\}$, $\hat{g}'_i := \mathsf{PolyQuotient}(\hat{g}_i, \mathcal{G}_i)$, and $\mathsf{Fill}_i(z_{i,j}) := \hat{g}'_i(z_{i,j})$ (if $z_{i,j} \in \mathcal{L}_i$).

     Additionally, the honest prover defines the degree-corrected polynomial $\hat{f}_i \in \mathbb{F}^{<d_i}[X]$ as follows:
     $$\hat{f}_i := \mathsf{DegCor}(d_i, \gamma_i, \hat{g}'_i, d_i - |\mathcal{G}_i|)\,.$$
     The protocol proceeds to the next iteration with $\hat{f}_i$.
- **Final round:** The prover sends d_M coefficients of a polynomial $\hat{p} \in \mathbb{F}^{<d_M}[X]$. In the honest case, $\hat{p} := \mathsf{Fold}(\hat{f}_M, k_M, \alpha_M)$.
- **Verifier decision phase**:
  1. **Main loop:** For $i = 1, \ldots, M$:
     (a) For every $j \in [t_{i-1}]$, query $\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1})$ at $z_{i,j}$. This involves querying f_{i-1} at all k_{i-1} points $x \in \mathcal{L}_{i-1}$ with $x^{k_{i-1}} = z_{i,j}$.
     (b) Define $\mathcal{G}_i := \{r^{\text{out}}_{i,1}, \ldots, r^{\text{out}}_{i,s}, z_{i,1}, \ldots, z_{i,t_{i-1}}\}$, and let $\mathsf{Ans}_i \colon \mathcal{G}_i \to \mathbb{F}$ be the function where $\mathsf{Ans}_i(r^{\text{out}}_{i,j}) = \beta_{i,j}$ and $\mathsf{Ans}_i(z_{i,j}) = \mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1})(z_{i,j})$. Finally, (virtually) set $g'_i := \mathsf{Quotient}(g_i, \mathcal{G}_i, \mathsf{Ans}_i, \mathsf{Fill}_i)$.
     (c) Define the virtual oracle $f_i \colon \mathcal{L}_i \to \mathbb{F}$ as follows:
     $$f_i := \mathsf{DegCor}(d_i, \gamma_i, g'_i, d_i - |\mathcal{G}_i|)\,.$$
     Observe that a query x to f_i translates to a single query either to g_i (if $x \notin \mathcal{G}_i$) or to Fill_i (if $x \in \mathcal{G}_i$).

2. **Consistency with final polynomial:**
   (a) Sample random points $r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}} \leftarrow \mathcal{L}_M^{k_M}$.
   (b) Check that $\hat{p}(r_j^{\mathsf{fin}}) = \mathsf{Fold}(f_M, k_M, \alpha_M)(r_j^{\mathsf{fin}})$ for every $j \in [t_M]$.
3. **Consistency with** Ans**:** For every $i \in \{1, \ldots, M\}$ and every $x \in \mathcal{G}_i \cap \mathcal{L}_i$ query $g_i(x)$ and check that $g_i(x) = \mathsf{Ans}_i(x)$.

Remark 5.4.3. If $\mathcal{L}_{i-1}^{k_{i-1}} \cap \mathcal{L}_i = \emptyset$ for every $i \in [M]$ then the oracles Fill_i and the verifier's check in Item 3 can be removed, reducing the proof length to $M \cdot s + \frac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} |\mathcal{L}_i|$ and query complexity to $\sum_{i=1}^{M} t_i$.

Complexity parameters. We analyze the complexity measures of Theorem 5.4.2.

- *Rounds.* The protocol has $2 \cdot M + 1$ rounds.
- *Proof length.* In iteration $i \in \{1, \ldots, M\}$ the prover sends g_i, of length $|\mathcal{L}_i|$, out-of-domain replies $\beta_{i,1}, \ldots, \beta_{i,s}$, and the Fill_i function, of length at most $t_{i-1} + s$. In the final round, the prover sends $d_M := d / \prod_{i=0}^{M} k_i$ field elements. Thus the oracle proof length is $\sum_{i=1}^{M} (|\mathcal{L}_i| + t_{i-1} + s)$ and the number of field elements sent is $M \cdot s + d / \prod_{i=0}^{M} k_i$. Therefore the total proof length is: $2 \cdot M \cdot s + \frac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} (|\mathcal{L}_i| + t_{i-1})$.
- *Input query complexity.* The verifier reads k_0 points t_0 times. Since each set of k_0 points are always queried together, they can be grouped together into a single symbol. The input query complexity over this alphabet is t_0.
- *Proof query complexity.* For $i \in \{1, \ldots, M\}$, the verifier performs t_i queries to $\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1})$, which induces reading k_{i-1} symbols from f_{i-1}. The verifier also queries f_i at at most $|\mathcal{G}_i| - 1 \leq t_i$ points. If $i = 1$ then this is where things end, but when $i > 1$, f_{i-1} is a virtual function where every query maps to a single query to either g_{i-1} or to Fill_{i-1}. Thus the verifier queries g_{i-1} at k_{i-1} points per query. Since these k_i symbols are always read together, they can be grouped together into a single symbol of a larger alphabet. Therefore the query complexity to the proof strings over this alphabet is $2 \cdot \sum_{i=1}^{M} t_i$.

5.4.2 Round-by-round soundness

We analyze the round-by-round soundness of STIR.

Lemma 5.4.4. *Consider* $(\mathbb{F}, M, d, k_0, \ldots, k_M, \mathcal{L}_0, \ldots, \mathcal{L}_M, t_0, \ldots, t_M, s)$ *and* $d_0, \ldots, d_M$ *as in Theorem 5.4.2, and for every* $0 \leq i \leq M$ *let* $\rho_i := d_i / |\mathcal{L}_i|$. *For every* $f \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0]$ *and every* $\delta_0, \ldots, \delta_M$ *where*

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1 - \mathsf{B}(\rho_0))$,
- *for every* $0 < i \leq M$: $\delta_i \in (0,\ \min\{1 - \rho_i - 1/|\mathcal{L}_i|,\ 1 - \mathsf{B}(\rho_i)\})$, *and*

- *for every* $0 < i \leq M$: $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]$ *is* (δ_i, ℓ_i)*-list decodable,*

STIR (Theorem 5.4.2) has round-by-round soundness error $(\varepsilon^{\mathsf{fold}}, \varepsilon_1^{\mathsf{out}}, \varepsilon_1^{\mathsf{shift}}, \ldots, \varepsilon_M^{\mathsf{out}}, \varepsilon_M^{\mathsf{shift}}, \varepsilon^{\mathsf{fin}})$ *where:*

- $\varepsilon^{\mathsf{fold}} \leq \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0)$.
- $\varepsilon_i^{\mathsf{out}} \leq \frac{\ell_i^2}{2} \cdot \left(\frac{d_i}{|\mathbb{F}| - |\mathcal{L}_i|}\right)^s$.
- $\varepsilon_i^{\mathsf{shift}} \leq (1 - \delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i)$.
- $\varepsilon^{\mathsf{fin}} \leq (1 - \delta_M)^{t_M}$.

Above, B *and* $\mathsf{err}^\star$ *are the proximity bound and error (respectively) described in Section 5.3.1.*

Proof. Establishing round-by-round soundness requires defining a state function, which in turn requires specifying in more detail the structure of an interaction transcript for STIR. We also discuss how to derive a function f_{i-1} from f, the main function being tested and a partial transcript for $i-1$ full iterations of STIR.

In the initial round, the transcript is empty, and we can trivially derive $f_0 := f$. In subsequent rounds, the transcript has the form

$$\mathsf{tr} := (\alpha_0, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, \mathsf{tr}') \,,$$

where $\mathsf{tr}_j := (r_j^{\mathsf{out}}, \beta_j, (\alpha_j, \gamma_j, z_{j,1}, \ldots, z_{j,t_j}))$ is a full transcript of the j-th iteration of STIR and tr' is a partial transcript of iteration i (or is equal to $\hat{p}$ in the case that we are in the final round). Given such a transcript and the function $f_0 := f$ we derive a function f_{i-1} in an identical way to the virtual function defined by the verifier algorithm, by running the main loop (Item 1) for $i-1$ times.

We now describe the state function State and analyze its error.

0. **State function for empty transcript.** Given a function $f \colon \mathcal{L} \to \mathbb{F}$ we set $\mathsf{State}(f, \emptyset) = 1$ if and only if $f \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$.

1. **Bounding** $\varepsilon^{\mathsf{fold}}$**.** The interaction starts with the verifier sending α_0.

   - *State function.* We set $\mathsf{State}(f, \alpha_0) = 1$ if and only if

   $$\Delta(\mathsf{Fold}(f, k_0, \alpha_0), \mathsf{RS}[\mathbb{F}, \mathcal{L}^{k_0}, d_0/k_0]) \leq \delta_0 \,.$$

   - *Bounding the error.* Since $\delta_0 < 1 - \mathsf{B}(\rho_0)$, from Theorem 5.3.9 we obtain that

   $$\begin{aligned} \varepsilon^{\mathsf{fold}} &= \Pr_{\alpha_0}\left[\mathsf{State}(f, \alpha_0) = 1 \,\middle|\, \mathsf{State}(f, \emptyset) = 0\right] \\ &= \Pr_{\alpha_0 \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k_0, \alpha_0), \mathsf{RS}[\mathbb{F}, \mathcal{L}^{k_0}, d_0/k_0]) \leq \delta_0\right] \\ &< \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0) \,. \end{aligned}$$

2. **Bounding $\varepsilon_i^{\mathsf{out}}$.** The transcript so far has the form $\mathsf{tr} := (\alpha_0, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, g_i)$, and the verifier sends randomness $r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}}$. Deriving f_{i-1} from tr, we define the state function.

   - *State function.* We set $\mathsf{State}(f, \mathsf{tr}||(r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}})) = 1$ if both conditions below hold.
     (a) At least one of the following holds:
        i. $\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_i]) \leq \delta_{i-1}$, or
        ii. there exist distinct codewords $u, u' \in \Lambda(g_i, d_i, \delta_i)$ such that $\hat{u}(r_{i,j}^{\mathsf{out}}) = \hat{u}'(r_{i,j}^{\mathsf{out}})$ for every $j \in [s]$.
     (b) For every $j \in \{1, \ldots, i-1\}$, $g_j(x) = \mathsf{Ans}_j(x)$ for every $x \in \mathcal{L}_j \cap \mathcal{G}_j$ where Ans_j is defined as in the verifier decision algorithm.
   - *Bounding the error.* We show that

$$\varepsilon_i^{\mathsf{out}} = \Pr_{r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}}} \left[\mathsf{State}(f, \mathsf{tr}||(r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \right] \leq \frac{\ell_i^2}{2} \cdot \left(\frac{d_i}{|\mathbb{F}| - |\mathcal{L}_i|} \right)^s .$$

     Suppose that $\mathsf{State}(f, \mathsf{tr}) = 0$. If there exists $j \in \{1, \ldots, i-1\}$ and $x \in \mathcal{L}_j \cap \mathcal{G}_j$ such that $g_j(x) \neq \mathsf{Ans}_j(x)$ then Item 2b cannot hold. Thus in order for the state to switch to 1, we assume Item 2b holds. By the assumption that $\mathsf{State}(f, \mathsf{tr}) = 0$:

$$\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_{i-1}/k_{i-1}]) > \delta_{i-1} .$$

     This, together with the fact that $d_i := d_{i-1}/k_{i-1}$, rules out Item 2(a)i. Hence we only need to bound the probability that Item 2(a)ii holds; by Theorem 5.3.5 this probability is at most $\frac{\ell_i^2}{2} \cdot \left(\frac{d_i}{|\mathbb{F}| - |\mathcal{L}_i|} \right)^s$.

3. **Bounding $\varepsilon_i^{\mathsf{shift}}$.** The transcript so far has the form $\mathsf{tr} := (\alpha_0, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, g_i, r_i^{\mathsf{out}}, \beta_i)$ and the verifier sends $\alpha_i, \gamma_i, z_1, \ldots, z_{t_{i-1}}$. Deriving f_{i-1} from tr, we define the state function.

   - *State function.* We set $\mathsf{State}(f, \mathsf{tr}||(\alpha_i, \gamma_i, z_1, \ldots, z_{t_{i-1}})) = 1$ if and only if both of the following hold:
     (a) $\Delta(\mathsf{Fold}(f_i, k_i, \alpha_i), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i^{k_i}, d_i/k_i]) \leq \delta_i$ where f_i is defined using f_{i-1}, g_i, $(r_{i,j}^{\mathsf{out}}, \beta_{i,j})_{j \in [s]}$, γ_i, and $z_{i,1}, \ldots, z_{i,t}$ as in Item 1c in the verifier decision algorithm;
     (b) for every $j \in \{1, \ldots, i\}$, $g_j(x) = \mathsf{Ans}_j(x)$ for every $x \in \mathcal{L}_j \cap \mathcal{G}_j$ where Ans_j is defined as in the verifier decision algorithm.
   - *Bounding the error.* We show that

$$\begin{aligned} \varepsilon_i^{\mathsf{shift}} &= \Pr_{\substack{\alpha_i, \gamma_i, \\ z_1, \ldots, z_{t_{i-1}}}} \left[\mathsf{State}(f, \mathsf{tr}||(\alpha_i, \gamma_i, z_1, \ldots, z_{t_{i-1}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \right] \\ &\leq (1 - \delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) . \end{aligned}$$

Suppose that $\mathsf{State}(f, \mathsf{tr}) = 0$. If this is due to the fact that there exists $j \in \{1, \ldots, i-1\}$ and $x \in \mathcal{L}_j \cap \mathcal{G}_j$ such that $g_j(x) \neq \mathsf{Ans}_j(x)$ (i.e., Item 2b does not hold), then Item 3b does not hold. Hence we assume that this is not the case. We show that except with probability $(1-\delta_i)^{t_i}$ there are no codewords that are close to g_i whose low-degree extensions agree on the points where we will later quotient g_i.

Claim 5.4.5. *With probability $1-(1-\delta_{i-1})^{t_{i-1}}$ over the choice of $z_{i,1}, \ldots, z_{i,t_{i-1}}$ for every $u \in \Lambda(g_i, d_i, \delta_i)$ there exists $x \in \mathcal{G}_i$ such that $\hat{u}(x) \neq \mathsf{Ans}_i(x)$.*

Proof. Since $\mathsf{State}(f, \mathsf{tr}) = 0$, by Item 2(a)ii there is at most one $u \in \Lambda(g_i, d_i, \delta_i)$ such that $\hat{u}(r^{\mathsf{out}}_{i,j}) = \beta_{i,j}$ for every $j \in [s]$. If no such codeword exists, then the claim holds trivially since $\mathsf{Ans}_i(r^{\mathsf{out}}_{i,j}) = \beta_{i,j}$ for some j. We are left to analyze the case where there is exactly one codeword $u \in \Lambda(g_i, d_i, \delta_i)$ for which $\hat{u}(r^{\mathsf{out}}_{i,j}) = \beta_{i,j}$ for every j.
Since $\mathsf{State}(f_0, \mathsf{tr}) = 0$ we have that

$$\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1}), \hat{u}(\mathcal{L}_{i-1}^{k_{i-1}})) \geq \Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_i]) > \delta_{i-1}\,.$$

Thus, for every j the probability that $\hat{u}(z_{i,j}) \neq \mathsf{Fold}(f_{i-1}, k_{i-1}, \alpha_{i-1})(z_{i,j}) = \mathsf{Ans}_i(z_{i,j})$ is at least δ_{i-1}. It follows that the probability that this event does not occur for every $j \in [t_{i-1}]$ simultaneously is at most $(1-\delta_{i-1})^{t_{i-1}}$. □

Suppose that event described in Theorem 5.4.5 occurs. Then by Theorem 5.3.4:

$$\Delta\left(g_i', \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i - |\mathcal{G}_i|]\right) + \frac{|\{x \in \mathcal{G}_i : g_i(x) \neq \mathsf{Ans}_i(x)\}|}{|\mathcal{L}_i|} > \delta_i\,.$$

Due to Item 3b, in order for the state to become 1, it must be that $g_i(x) = \mathsf{Ans}_i(x)$ for every $x \in \mathcal{L}_i \cap \mathcal{G}_i$, and so we conclude that $\Delta\left(g_i', \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i - |\mathcal{G}_i|]\right) > \delta_i$. Recalling that $\delta_i < \min\{1 - \mathsf{B}(\rho_i), 1 - \rho_i - 1/|\mathcal{L}_i|\}$ and using Theorem 5.3.13 we deduce that

$$\Pr_{\gamma_i \leftarrow \mathbb{F}}\left[\Delta\left(\mathsf{DegCor}(d_i, \gamma_i, g_i', d_i - |\mathcal{G}_i|), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]\right) \leq \delta_i\right] \leq \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s)\,.$$

Finally, observe that $f_i := \mathsf{DegCor}(d_i, \gamma_i, g_i', d_i - |\mathcal{G}_i|)$ and that if $\Delta\left(f_i, \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]\right) \leq \delta_i$, then by Theorem 5.3.9:

$$\Pr_{\alpha_i \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f_i, k_i, \alpha_i), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i^{k_i}, d_i/k_i]) \leq \delta_i\right] \leq \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i)\,.$$

Putting all of the above probabilities together we conclude that

$$\varepsilon_i^{\mathsf{shift}} \leq (1-\delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i)\,.$$

4. **Bounding ε^{fin}:** At this stage the transcript holds the form $\mathsf{tr} := (\alpha_0, \mathsf{tr}_1, \ldots, \mathsf{tr}_M, \hat{p})$. The verifier chooses points $(r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})$. Deriving f_M from the transcript, we define the state function.

   - *State function.* We set $\mathsf{State}(f, \mathsf{tr}||(r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})) = 1$ if and only if both of the following hold:
     (a) $\hat{p}(r_j^{\text{fin}}) = \mathsf{Fold}(f_M, k, \alpha_M)(r_j^{\text{fin}})$ for every $j \in [t_i]$.
     (b) For every $0 < j \leq M$: $g_j(x) = \mathsf{Ans}_j(x)$ for every $x \in \mathcal{L}_j \cap \mathcal{G}_j$ where Ans_j is defined as in the verifier decision algorithm.
   - *Bounding the error.* We show that

$$\varepsilon^{\text{fin}} = \Pr_{r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}}} \left[\mathsf{State}(f, \mathsf{tr}||(r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \right] \leq (1 - \delta_M)^{t_M} .$$

     Suppose that $\mathsf{State}(f, \mathsf{tr}) = 0$. If Item 3b does not hold, then Item 4b also does not hold, and so the state cannot change to 1. Assuming this is not the case, it follows from Item 3a that:

$$\Delta(\mathsf{Fold}(f_M, k_M, \alpha_M), \hat{p}(\mathcal{L}_M^{k_M})) \geq \Delta(\mathsf{Fold}(f_M, k_M, \alpha_M), \mathsf{RS}[\mathbb{F}, \mathcal{L}_M^{k_M}, d_M]) > \delta_M .$$

     Thus, for each j the probability that $\hat{p}(r_j^{\text{fin}}) = \mathsf{Fold}(f_M, k_M, \alpha_M)(r_j^{\text{fin}})$ is at most $1 - \delta_M$. It follows that the probability that this occurs for every $j \in [t_M]$ simultaneously is at most $(1 - \delta_M)^{t_M}$.

5. **Verifier decision.** If $\mathsf{State}(f, \mathsf{tr}) = 0$ for a full transcript tr, then the verifier rejects. This is due to the fact that the verifier checks in Item 2 that Item 4a holds, and in Item 3 that Item 4b holds. If $\mathsf{State}(f, \mathsf{tr}) = 0$ then one of the two must not hold, and so the verifier rejects.

□

5.4.3 Recommended parameters

We provide recommended parameters for STIR for achieving round-by-round soundness error $2^{-\lambda}$. These parameters use the same folding parameter $k > 2$ for every round, and enforce that the evaluation domain shrinks by a multiplicative factor of 2 in every round, so that the total proof length is linear for reasonable security parameters. We begin by describing parameter settings for provable soundness, and then describe settings assuming a list-decoding conjecture, which allows for smaller constants and better concrete efficiency. For ease readability, detailed derivations are deferred to Section 5.9.

Ingredients. The protocol receives the following ingredients: (a) A Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ with rate $\rho := d/|\mathcal{L}|$ where $d \geq 4$ is a power of two, and $\mathcal{L}$ is a smooth domain. (b) A security parameter $\lambda \in \mathbb{N}$. (c) A stopping degree $d_{\text{stop}} \in \mathbb{N}$

where d_{stop} is a power of two such that $d \geq d_{\mathsf{stop}} \geq 4$. (d) A proximity parameter $\delta \in (0,1)$. (e) A folding parameter $k \in \mathbb{N}$ that is a power of two where $k \geq 4$.

Settings. We plug these parameters into Theorem 5.4.2 and set the following: (a) $M := \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor$. (b) $k_i := k$. (c) $s := 1$. (d) $\mathcal{L}_i = \omega \cdot \langle \omega^2 \rangle$ where ω is the generator of $\mathcal{L}_{i-1}$.[8] (e) $\eta_0 = \left(\frac{2^\lambda \cdot (k-1) \cdot (d/k)^2}{2^7 \cdot |\mathbb{F}|}\right)^{1/7}$, $t_0 := \left\lceil \frac{\lambda+1}{-\log(1-\delta')} \right\rceil$ where $\delta' := \min\{\delta, 1 - \sqrt{\rho} - \eta_0\}$ (f) for $0 < i < M$ set $d_i := d^{k^i}$, $\rho_i := (2/k)^i$,

$$\eta_i := \max\left\{ \left(\frac{2^\lambda \cdot d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)}\right)^{1/2}, \left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2\right)}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} \right\}.$$

and $t_i := \left\lceil \frac{\lambda+1}{-\log(\sqrt{\rho_i}+\eta_i)} \right\rceil$. (g) $t_M := \left\lceil \frac{\lambda}{-\log(\sqrt{\rho_M}+\eta_M)} \right\rceil$.

Resulting IOPP. If the field $\mathbb{F}$ satisfies

$$|\mathbb{F}| > 10^7 \cdot (\lambda+1) \cdot 2^{\lambda+1} \cdot d^2 \cdot |\mathcal{L}|^{3.5} \cdot \left(1 + \max\left\{ \left\lceil \frac{1}{-\log(1-\delta')} \right\rceil, \left\lceil \frac{1}{-\log(1.05 \cdot \sqrt{\rho})} \right\rceil \right\}\right),$$

then[9] the IOPP for $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ defined as above has the following properties:

- *Rounds:* $2 \cdot M + 1$.
- *Length:* $|\mathcal{L}| + O_k(d_{\mathsf{stop}} + \log(d/d_{\mathsf{stop}}))$.
- *Input queries:* $\left\lceil \frac{\lambda}{-\log(1-\delta')} \right\rceil$ where $\delta' := \min\{\delta, 1 - 1.05 \cdot \sqrt{\rho}\}$.
- *Proof queries:* $O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{-\log \rho}\right)\right)$.
- *Round-by-round soundness error:* $2^{-\lambda}$.

See Section 5.9.1 for a detailed derivation of these values.

Conjectured soundness. We use the following conjecture on the list-decoding properties of Reed–Solomon codes to improve the concrete parameters of STIR:

Conjecture 5.4.6 ([BGKS20; BCIKS20]). *Let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ be a Reed–Solomon code with rate $\rho := d/|\mathcal{L}|$. There exist constants $c_1, c_2, c_3 \in \mathbb{N}$ such that for every $\eta > 0$ and $0 < \delta < 1 - \rho - \eta$ the following hold:*

- *For functions $f_1 \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$, if*

$$\Pr_{r \leftarrow \mathbb{F}}\left[\Delta\left(\mathcal{C}, \mathsf{Combine}(r, (f_1, \ldots, f_\ell))\right) \leq \delta\right] > \mathsf{err}^\star(d, \rho, \delta, \ell) := \frac{(\ell-1)^{c_2} \cdot d^{c_2}}{\eta^{c_1} \cdot \rho^{c_1+c_2} \cdot |\mathbb{F}|}.$$

then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1-\delta) \cdot |\mathcal{L}|$, and

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

[8] Observe that, since k is a power of two, the sets $\mathcal{L}_{i-1}^k := \langle \omega^k \rangle$ and $\mathcal{L}_i := \omega \cdot \langle \omega^2 \rangle$ are disjoint as required for Theorem 5.4.3.

[9] Note that this bound is not tight.

- $\mathcal{C}$ *is* (δ, ℓ)*–list decodable for* $\ell \leq \left(\frac{d}{\rho \cdot \eta}\right)^{c_3}$.

The ingredients are identical as in the provable setting. The parameters in Theorem 5.4.2 are set identically, except that $s := 2$, $\eta_0 := \left(\frac{2^\lambda \cdot (k-1)^{c_2} \cdot (d/k)^{c_2}}{\rho^{c_1+c_2} \cdot |\mathbb{F}|}\right)^{1/c_1}$, and

$$\eta_i := \max \left\{ \begin{array}{c} \frac{2\cdot\rho_i}{d_i}, \frac{d_i}{\rho_i} \cdot \left(\frac{2^\lambda \cdot d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s}\right)^{\frac{1}{2\cdot c_3}}, \\ \left(\frac{2^{\lambda+1} \cdot d_i^{c_2}}{\rho_i^{c_1+c_2} \cdot |\mathbb{F}|} \cdot \left((t_{i-1} + s - 1)^{c_2} + \left(\frac{k-1}{k}\right)^{c_2}\right)\right)^{1/c_1} \end{array} \right\}.$$

Assuming Theorem 5.4.6, and the above with $c_1 = c_2 = c_3 = 1$, we have improved concrete parameters, and round-by-round soundness error $2^{-\lambda}$ when

$$|\mathbb{F}| > (\lambda + 1) \cdot 2^{\lambda+2} \cdot d \cdot |\mathcal{L}|^3 \cdot \left(s + \max\left\{\left\lceil \frac{1}{-\log(1-\delta')} \right\rceil, \left\lceil \frac{1}{-\log(1.5 \cdot \rho)} \right\rceil\right\}\right).$$

As in the provable parameter regime, this bound is not tight. See Section 5.9.2 for full derivations given this parameter setting.

5.5 Implementation and experimental results

We evaluate the performance of STIR in comparison to FRI [BBHR18], with regards to: (i) argument size; (ii) prover time; (iii) verifier time; and (iv) verifier hash complexity. Furthermore, we compare the argument sizes of STIR and FRI when they are used to realize a SNARK for R1CS.

5.5.1 Implementation

We implemented FRI and STIR in Rust, by leveraging the `arkworks` [ark] ecosystem for developing zkSNARKs. Our implementation and scripts are available at the repository `https://github.com/WizardOfMenlo/stir/`; later they will be open-sourced and integrated as part of `arkworks`.

Organization. We expose a common interface for low-degree testing, which we realize via FRI and STIR implementations. The interface is generic over the underlying (nice) field, the hash function used for the Merkle tree, and the (sponge) hash function used for the Fiat–Shamir transformation.

Cryptographic primitives. We use `arkworks` [ark] for several underlying cryptographic primitives. We use the crate `ark-ff` for field arithmetic, `ark-poly` for Fast Fourier Transforms, and `ark-crypto-primitives` for both the Merkle trees and sponges. We use the crates `sha3` and `blake3` for the hash functions used in Merkle trees and sponges. Our Poseidon parameter generation is according to the crate `poseidon-paramgen`, and the Poseidon implementation is again from the crate `ark-crypto-primitives`.

Optimizations. Our implementations of FRI and STIR should be considered reference implementations rather than optimized ones. We implemented optimizations such as path pruning for Merkle trees and reduced costly operations such as field inversion as much as possible. Nonetheless, we believe that there is room for further performance gains via additional optimizations. Further, we only performed partial parallelization of the prover algorithms.

SNARK for R1CS. We plan to implement the SNARK obtained by combining the PIOP in Section 5.8 with the compiler in Section 5.6. For now, the sizes that we report are obtained via a Python script.

5.5.2 Parameter choices

In our experiments, given a starting degree d and a rate ρ, we select parameters to instantiate both FRI and STIR for a Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$. We detail our parameter choice next.

Field and evaluation domain choice. We set $\mathbb{F}$ to be a 192-bit smooth prime field,[10] and let $\mathcal{L}$ be an arbitrary smooth domain with $|\mathcal{L}| = d/\rho$.

[10] We arbitrarily selected $\mathbb{F} = \mathbb{F}_p$ with $p = 2^{64} \cdot 259536638529657107390708680683681617371 + 1$.

Soundness. We target $\lambda = 128$ bits of security, by which we mean that for both STIR and FRI, we set the round-by-round soundness error of the IOPP to be 2^{-128} and let the hash function output used in the BCS transformation to have length 256 bits. To achieve round-by-round soundness error 2^{-128}, we use a proof-of-work of 22 bits. This means that when the prover (honest and malicious) performs the Fiat-Shamir query to derive randomness for the next round, the probability that it will "solve" the proof of work and get the desired randomness is only 2^{-22}. This increases the runtime of the honest prover; however, it reduces the round-by-round soundness required from the underlying IOP to only 2^{-106}. The result is a smaller number of queries and, thus, a smaller argument size. This optimization is performed in both the FRI and STIR implementations.

Repetitions. The number of repetitions in FRI and in STIR to achieve round-by-round soundness error $2^{-\lambda_{\text{prot}}}$ was selected by assuming Theorem 5.4.6 with $c_1 = c_2 = c_3 = 1$ and, in both protocols, discounting the negligible security deficit imposed by η. For STIR we set $s_i = 2$ for every iteration since this was more than enough to achieve 128 bits of security with the field $\mathbb{F}$.

Folding parameter and stopping conditions. When selecting the folding parameter k, we observed empirically that using a folding-factor $k = 16$ minimizes the argument size of STIR, while $k = 8$ minimizes that of FRI. In both STIR and FRI, we stop the protocol when the final degree d_M is at most 2^6.

Compiling into a SNARG. When compiling the IOPP into a SNARG via the transformation in [BCS16], we need to choose a hash function for the Merkle tree and a hash function for the Fiat–Shamir transformation. We consider two configurations:

- *Primary configuration (Native).* We use BLAKE3 as the hash function for the Fiat–Shamir transformation, and SHA3 for the Merkle trees. This is the primary configuration we measure and discuss in this document.

- *Secondary configuration (Algebraic).* We use BLAKE3 as the hash function for the Fiat–Shamir transformation and the Poseidon algebraic hash function [GKKRRS19] for the Merkle trees.

When used to instantiate a SNARK as in Section 5.8, the former configuration would typically be used for direct instantiation, while the latter would be used to construct SNARKs that are then recursively proved and verified (due to the algebraic structure of Poseidon).

5.5.3 Benchmarks

We ran our benchmarks on an AWS `r6a.24xlarge` instance with 96 vCPU and 768 GiB of memory (AMD EPYC 7R13 Processor @ 2.65 GHz), and compiled using `rustc 1.77.0-nightly`. Our methodology is the following. We first select a degree-rate pair (d, ρ).

- *Primary configuration.* We select a degree $d \in \{2^{18}, 2^{20}, 2^{22}, 2^{24}, 2^{26}, 2^{28}, 2^{30}\}$ and a rate $\rho \in \{1/2, 1/4, 1/8, 1/16\}$. We ignore the rate-degree pair $(d, \rho) = (2^{30}, 1/16)$, as our instance ran out of memory while running the argument prover.
- *Secondary configuration.* We select a degree $d \in \{2^{18}, 2^{20}, 2^{22}, 2^{24}, 2^{26}, 2^{28}\}$ and rate $\rho \in \{1/2, 1/4, 1/8\}$.

Having chosen (d, ρ), we select parameters as detailed in Section 5.5.2. Given those parameters, we benchmark both the argument prover and argument verifier, collecting: (i) argument size; (ii) prover time; (iii) verifier time; and (iv) verifier hash complexity.

All of our experiments using the *native* configuration were run serially. Those using the *algebraic* configuration instead generate argument strings in parallel to speed up the benchmarking process, but, since we did not parallelize optimally our FRI prover implementation, we do not include those measured prover times for fairness.

5.5.4 Results

We discuss the results of our experiments with the *native* configuration. Figure 5.2 graphically compares STIR and FRI for rate $1/2$ and varying degrees. Table 5.2 contains all experimental data for the native configuration. Additional experimental results are in Section 5.7. We discuss each metric individually, arbitrarily focusing on the case of degree $d = 2^{24}$ and rate $\rho = 1/2$.

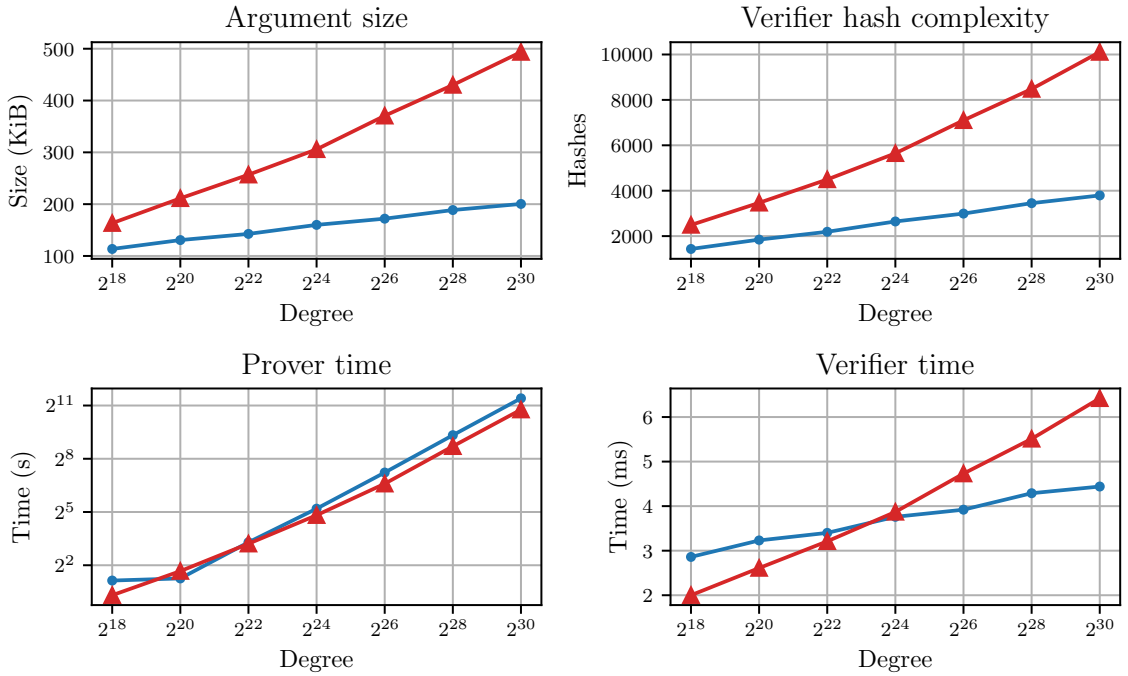


Figure 5.2: Comparison of FRI and STIR for $\rho = 1/2$. FRI: ▲, STIR: ●. Lower is better. Note that prover time is displayed with logarithmic scaling.

Argument size. Across all degrees and rates, STIR's arguments are significantly smaller than FRI's. The improvement ranges from 1.25× to 2.46×, and it is more pronounced for larger degrees and rates. For $d = 2^{24}$ and rate $\rho = 1/2$, STIR's argument size is 160 KiB whereas FRI's is 306 KiB.

Verifier time and hash complexity. Across all degrees and rates, STIR's hash complexity is significantly smaller than FRI, with STIR's verifier performing between 1.55× to 2.67× fewer hashes than FRI's. Again, this relative improvement is more evident with larger degrees and rates. As for verifier time, relative comparison is more nuanced, with the relative difference ranging from a 0.7× slowdown to a 1.45× speedup. For larger degrees and larger rate, STIR's verifier performs better than FRI's, while for smaller degrees the added algebraic complexity results in a slowdown.

For $d = 2^{24}$ and rate $\rho = 1/2$, STIR's verifier performs 2645 hashes and runs in 3.8ms whereas the FRI verifier performs 5647 hashes and runs in 3.9ms.

Prover time. STIR's prover time is slightly larger than FRI on our test set,[11] with the slowdown ranging from 0.64× to 0.95×. Concretely, for $d = 2^{24}$ and rate $1/2$, generating a proof (serially) takes STIR 36s and FRI 28s. This slowdown decreases when the rate decreases, as then the cost of the initial FFT (shared between STIR and FRI) accounts for a larger portion of the prover's running time.

Algebraic configuration. As expected, the results of the *algebraic* configuration for argument size and verifier hashes are in line with those observed in the *native* configuration. As is evidenced by the experimental results described in Table 5.4, since the relative cost of performing hashes is higher in this configuration, we observe that STIR's verifier now outperforms FRI's across all settings tested. We measure this speedup to be between 1.48× to 2.34×.

SNARK for R1CS. We computed the argument size for both STIR and FRI when used to instantiate a SNARK for R1CS using an idealized Python script. We used the *native* configuration for both. We detail the result in Table 5.5. The improvement in argument size of STIR over FRI translates to this setting, with STIR-based SNARKs between 1.29× to 2.25× smaller than their FRI-based counterparts. Concretely, for instances of size $n = 2^{24}$ with rate $1/2$, STIR-based SNARKs have size 220 KiB whereas FRI has size 422 KiB.

Discussion.

- The asymptotic improvement in query complexity of STIR over FRI translates in concrete and significant improvements in both argument size and verifier hash complexity across *all configurations and parameters tested*.
- As for verifier time, in the native configuration the added algebraic complexity of STIR results in a slower verifier for small degrees and rates, while when those are large STIR's verifier outperforms FRI's. In the algebraic configuration instead,

[11] The table might seem to suggest that for small degrees STIR's prover can be faster than FRI's. In fact, this is due to proof-of-work taking a sizable portion of the prover computational cost for small degrees, and the high variance of this operation.

ρ \ d	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
	Argument size $\left(\text{KiB}, \frac{\text{FRI}}{\text{STIR}}\right)$						
$1/2$	$\frac{163}{114} \approx 1.44\times$	$\frac{211}{131} \approx 1.62\times$	$\frac{257}{143} \approx 1.8\times$	$\frac{306}{160} \approx 1.91\times$	$\frac{371}{172} \approx 2.15\times$	$\frac{430}{189} \approx 2.28\times$	$\frac{494}{200} \approx 2.46\times$
$1/4$	$\frac{99}{73} \approx 1.34\times$	$\frac{129}{87} \approx 1.48\times$	$\frac{154}{94} \approx 1.63\times$	$\frac{177}{107} \approx 1.66\times$	$\frac{211}{114} \approx 1.84\times$	$\frac{249}{128} \approx 1.95\times$	$\frac{277}{136} \approx 2.04\times$
$1/8$	$\frac{76}{58} \approx 1.32\times$	$\frac{96}{69} \approx 1.39\times$	$\frac{118}{75} \approx 1.57\times$	$\frac{134}{86} \approx 1.55\times$	$\frac{157}{93} \approx 1.7\times$	$\frac{184}{104} \approx 1.77\times$	$\frac{204}{110} \approx 1.85\times$
$1/16$	$\frac{62}{50} \approx 1.25\times$	$\frac{77}{61} \approx 1.27\times$	$\frac{95}{66} \approx 1.44\times$	$\frac{107}{76} \approx 1.41\times$	$\frac{127}{82} \approx 1.56\times$	$\frac{147}{92} \approx 1.6\times$	-
	Verifier time $\left(\text{ms}, \frac{\text{FRI}}{\text{STIR}}\right)$						
$1/2$	$\frac{2.0}{2.9} \approx 0.7\times$	$\frac{2.6}{3.2} \approx 0.81\times$	$\frac{3.2}{3.4} \approx 0.94\times$	$\frac{3.9}{3.8} \approx 1.03\times$	$\frac{4.7}{3.9} \approx 1.21\times$	$\frac{5.5}{4.3} \approx 1.28\times$	$\frac{6.4}{4.4} \approx 1.45\times$
$1/4$	$\frac{1.2}{1.7} \approx 0.74\times$	$\frac{1.6}{2.0} \approx 0.8\times$	$\frac{1.9}{2.1} \approx 0.92\times$	$\frac{2.3}{2.4} \approx 0.97\times$	$\frac{2.7}{2.5} \approx 1.09\times$	$\frac{3.2}{2.8} \approx 1.17\times$	$\frac{3.7}{2.9} \approx 1.27\times$
$1/8$	$\frac{1.0}{1.2} \approx 0.78\times$	$\frac{1.2}{1.5} \approx 0.79\times$	$\frac{1.5}{1.6} \approx 0.93\times$	$\frac{1.8}{1.9} \approx 0.95\times$	$\frac{2.1}{2.0} \approx 1.06\times$	$\frac{2.4}{2.2} \approx 1.11\times$	$\frac{2.8}{2.3} \approx 1.22\times$
$1/16$	$\frac{0.8}{1.0} \approx 0.79\times$	$\frac{1.0}{1.3} \approx 0.76\times$	$\frac{1.2}{1.4} \approx 0.89\times$	$\frac{1.4}{1.6} \approx 0.89\times$	$\frac{1.7}{1.7} \approx 1.01\times$	$\frac{2.0}{1.9} \approx 1.04\times$	-
	Verifier hashes $\left(\frac{\text{FRI}}{\text{STIR}}\right)$						
$1/2$	$\frac{2490}{1434} \approx 1.74\times$	$\frac{3466}{1846} \approx 1.88\times$	$\frac{4494}{2191} \approx 2.05\times$	$\frac{5647}{2645} \approx 2.13\times$	$\frac{7100}{2992} \approx 2.37\times$	$\frac{8479}{3451} \approx 2.46\times$	$\frac{10107}{3792} \approx 2.67\times$
$1/4$	$\frac{1658}{1020} \approx 1.63\times$	$\frac{2270}{1329} \approx 1.71\times$	$\frac{2821}{1521} \approx 1.85\times$	$\frac{3459}{1849} \approx 1.87\times$	$\frac{4220}{2050} \approx 2.06\times$	$\frac{5072}{2401} \approx 2.11\times$	$\frac{5885}{2622} \approx 2.24\times$
$1/8$	$\frac{1374}{843} \approx 1.63\times$	$\frac{1801}{1098} \approx 1.64\times$	$\frac{2258}{1256} \approx 1.8\times$	$\frac{2720}{1534} \approx 1.77\times$	$\frac{3271}{1697} \approx 1.93\times$	$\frac{3879}{2010} \approx 1.93\times$	$\frac{4455}{2172} \approx 2.05\times$
$1/16$	$\frac{1185}{765} \approx 1.55\times$	$\frac{1518}{1014} \approx 1.5\times$	$\frac{1898}{1147} \approx 1.65\times$	$\frac{2233}{1376} \approx 1.62\times$	$\frac{2718}{1537} \approx 1.77\times$	$\frac{3166}{1792} \approx 1.77\times$	-
	Prover time $\left(\text{s}, \frac{\text{FRI}}{\text{STIR}}\right)$						
$1/2$	$\frac{1.2}{2.2} \approx 0.56\times$	$\frac{3.2}{2.4} \approx 1.33\times$	$\frac{9.3}{9.8} \approx 0.95\times$	$\frac{28}{36} \approx 0.77\times$	$\frac{97}{150} \approx 0.65\times$	$\frac{420}{640} \approx 0.65\times$	$\frac{1700}{2700} \approx 0.64\times$
$1/4$	$\frac{2.3}{1.1} \approx 2.11\times$	$\frac{2.7}{3.9} \approx 0.7\times$	$\frac{11}{14} \approx 0.81\times$	$\frac{47}{58} \approx 0.8\times$	$\frac{200}{250} \approx 0.8\times$	$\frac{860}{1100} \approx 0.78\times$	$\frac{3600}{4800} \approx 0.75\times$
$1/8$	$\frac{1.4}{1.9} \approx 0.73\times$	$\frac{5.4}{6.1} \approx 0.89\times$	$\frac{22}{26} \approx 0.86\times$	$\frac{93}{110} \approx 0.85\times$	$\frac{400}{480} \approx 0.83\times$	$\frac{1700}{2100} \approx 0.8\times$	$\frac{7000}{8900} \approx 0.79\times$
$1/16$	$\frac{2.7}{2.9} \approx 0.93\times$	$\frac{10}{11} \approx 0.93\times$	$\frac{44}{48} \approx 0.93\times$	$\frac{190}{210} \approx 0.88\times$	$\frac{780}{930} \approx 0.84\times$	$\frac{3300}{4100} \approx 0.82\times$	-

Table 5.2: Comparison of concrete costs between STIR and FRI. The numerator of the fraction is the cost associated to FRI, while the denominator is that associated to STIR. For all metrics, lower is better.

STIR's verifier is significantly faster than FRI's across all parameters tested, thanks to its reduced hash complexity.

- FRI maintains an edge over STIR in terms of prover time. This is mostly due to the fact that STIR's prover performs an FFT per round, while FRI's prover, once the initial FFT is computed, runs in linear time. When the cost of this initial FFT (which is shared between FRI and STIR) increases (i.e. on larger rates), STIR's prover compares more favorably with FRI's.
- When used within a larger SNARK, STIR's argument size reduction over FRI's result in arguments that are overall much smaller across all parameters that we computed.

The rate offers a trade-off between prover time and argument size: one can decrease the rate, thus increasing prover running time while reducing argument size. Since the FRI prover is faster than STIR, we ask whether FRI outperforms STIR in argument size when used with smaller rate. We show that experimentally, this is not the case. Two examples are shown in Table 5.3.

	$d = 2^{24}$			$d = 2^{28}$		
	STIR $\rho = 1/4$	FRI $\rho = 1/8$	Ratio	STIR $\rho = 1/4$	FRI $\rho = 1/8$	Ratio
Argument size	107 KiB	134 KiB	1.25×	128 KiB	184 KiB	1.44×
Verifier time	2.4 ms	1.8 ms	0.75×	2.8 ms	2.4 ms	0.85×
Verifier hashes	1849	2720	1.47×	2401	3879	1.61×
Prover time	58 s	93 s	1.60×	1100 s	1700 s	1.54×

Table 5.3: Comparison of STIR and FRI on different rates.

5.6 An efficient compiler for poly-IOPs

We describe a transformation that combines a poly-IOP and an RS-code IOPP to obtain a corresponding IOP. This transformation is concretely efficient, and has round-by-round knowledge soundness related to the round-by-round knowledge soundness of the poly-IOP.

Theorem 5.6.1. *Consider the following ingredients:*

- *A poly-IOPP* $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$ *for a relation* $\mathcal{R}$ *with round complexity* $\mathsf{k}_{\mathsf{poly}}$*, where in round* i $\mathbf{P}_{\mathsf{poly}}$ *sends* $(\hat{f}_{i,j} \in \mathbb{F}^{<d_{i,j}}[X])_{j\in[\ell_i]}$ *and the verifier makes at most* $\mathsf{q}_{i,j} < d_{i,j}$ *queries to* $\hat{f}_{i,j}$*. The poly-IOP has round-by-round knowledge soundness errors* $\mathsf{err}^{\mathsf{poly}}_1, \ldots, \mathsf{err}^{\mathsf{poly}}_{\mathsf{k}_{\mathsf{poly}}}$*.*
- *An IOPP* $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *for the code* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *(with rate* $\rho := d/|\mathcal{L}|$*) with round complexity* $\mathsf{k}_{\mathsf{prx}}$*, round-by-round soundness errors* $\mathsf{err}^{\mathsf{prx}}_1, \ldots, \mathsf{err}^{\mathsf{prx}}_{\mathsf{k}_{\mathsf{prx}}}$*, and* $\max_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}\{d_{i,j}\} \leq d$*.*
- *A proximity parameter* $\delta \in (0, 1 - \max\{\mathsf{B}(\rho), \rho + 1/|\mathcal{L}|\})$ *such that, for every* $i \in [\mathsf{k}_{\mathsf{poly}}]$*,* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i,j}]$ *is* $(\delta, \ell_{i,j})$*-list decodable.*
- *An extractor* $\mathbf{E}_{\mathsf{RS}}$ *that for every* i, j *list-decodes a codeword of distance at most* δ *from* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i,j}]$ *in time at most* $\mathsf{et}_{\mathsf{RS}}$*.*

Then there is an IOPP for $\mathcal{R}$ *with the following parameters:*[12]

	poly-IOPP for $\mathcal{R}$	IOPP for $\mathcal{C}$	$\rightarrow$ IOPP for $\mathcal{R}$
Rounds	$\mathsf{k}_{\mathsf{poly}}$	$\mathsf{k}_{\mathsf{prx}}$	$2\mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}} + 1$
Proof length	s_{poly}	$\mathsf{l}_{\mathsf{prx}}$	$\mathsf{l}_{\mathsf{prx}} + 2 \cdot \mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}} \cdot (\lvert\mathcal{L}\rvert + 2)$
Input queries	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$
Proof queries	$\mathsf{q}_{\mathsf{poly},\Pi}$	$\mathsf{q}_{\mathsf{prx},\Pi}$	$s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\Pi} + \mathsf{q}_{\mathsf{poly},\Pi}$
Verifier time	$\mathsf{vt}_{\mathsf{poly}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{vt}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \mathsf{q}_{i,j}^2)$
Extraction time	$\mathsf{et}_{\mathsf{poly}}$	-	$\mathsf{et}_{\mathsf{poly}} + s_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}}$

The compiled IOPP has round-by-round knowledge soundness error

$$(\varepsilon^{\mathsf{out}}_1, \varepsilon^{\mathsf{piop}}_1, \ldots, \varepsilon^{\mathsf{out}}_{\mathsf{k}_{\mathsf{poly}}}, \varepsilon^{\mathsf{piop}}_{\mathsf{k}_{\mathsf{poly}}}, \varepsilon^{\mathsf{com}}, \varepsilon^{\mathsf{prx}}_1, \ldots, \varepsilon^{\mathsf{prx}}_{\mathsf{k}_{\mathsf{prx}}}),$$

where (for inputs $(\mathbb{x}, \mathbb{y})$*):*

- $\varepsilon^{\mathsf{out}}_i \leq \frac{\sum_{j\in[\ell_i]} d_{i,j}\cdot\ell^2_{i,j}}{2\cdot|\mathbb{F}|}$.
- $\varepsilon^{\mathsf{piop}}_i \leq \mathsf{err}^{\mathsf{poly}}_i(\mathbb{x}, \mathbb{y})$.
- $\varepsilon^{\mathsf{com}} \leq \mathsf{err}^{\star}\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} (d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right)$.
- $\varepsilon^{\mathsf{prx}}_i \leq \mathsf{err}^{\mathsf{prx}}_i(\delta)$.

Above, B *and* $\mathsf{err}^{\star}$ *are the proximity bound and error (respectively) described in Section 5.3.1.*

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

5.6.1 Construction

We describe the transformation from a poly-IOPP to an IOPP, and then discuss its efficiency.

Construction 5.6.2. We construct an IOPP for $\mathcal{R}$ from the ingredients in Theorem 5.6.1.

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

1. **Poly-IOP interaction phase:**

   (a) For $i = 1, \ldots, \mathsf{k}_{\mathsf{poly}}$:

      i. **Poly-IOP prover message:** The prover sends oracle functions $(f_{i,j})_{j\in[\ell_i]}$ where $f_{i,j} \colon \mathcal{L} \to \mathbb{F}$. In the honest case, the prover computes $(\hat{f}_{i,j})_{j\in[\ell_i]} := \mathbf{P}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i-1})$, and sets $f_{i,j}$ to be the evaluation of $\hat{f}_{i,j}$ over $\mathcal{L}$.
      ii. **Out-of-domain sample:** The verifier sends $x_i \leftarrow \mathbb{F}$.
      iii. **Out-of-domain reply:** The prover sends field elements $(y_{i,j})_{j\in[\ell_i]}$. In the honest case $y_{i,j} := \hat{f}_{i,j}(x_i)$.
      iv. **Poly-IOP verifier message:** The verifier sends $\alpha_i \leftarrow \{0,1\}^{r_i}$.

2. **Low-degree test interaction phase:**

   (a) **Send query results and prepare for low-degree test:** The prover sends arrays of field elements $(A_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell]}$ where $|A_{i,j}| = \mathsf{q}_{i,j}$, and oracle functions $(w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell_i]}$ where $w_{i,j} \colon [\mathsf{q}_{i,j}+1] \to \mathbb{F}$.
   In the honest case the prover simulates the execution of

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

   For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$, set the following:

   - set $\mathcal{Q}_{i,j} \subseteq \mathbb{F}$ to be the set of queries made by $\mathbf{V}_{\mathsf{poly}}$ to $\hat{f}_{i,j}$;
   - set $\phi_{i,j} \colon [\mathsf{q}_{i,j}] \to \mathcal{Q}_{i,j}$ be the mapping where $\phi_{i,j}(k)$ returns the k-th query made to $\hat{f}_{i,j}$;
   - set $\mathcal{S}_{i,j} := \mathcal{Q}_{i,j} \cup \{x_i\}$ and $\hat{f}'_{i,j} := \mathsf{PolyQuotient}(\hat{f}_{i,j}, \mathcal{S}_{i,j})$.

   Then, the prover sets $A_{i,j}[k] := \hat{f}_{i,j}(\phi_{i,j}(k))$ and $w_{i,j}(k) := \hat{f}'_{i,j}(\phi_{i,j}(k))$ for $k < \mathsf{q}_{i,j}$, and $w_{i,j}(\mathsf{q}_{i,j}+1) := \hat{f}'_{i,j}(x_i)$.

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

(c) **Low-degree test:** Run the interaction phase of the low-degree test $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ for the code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$. The honest prover acts according to the polynomial $\hat{g} \in \mathbb{F}^{<d}[X]$ defined as:

$$\hat{g} := \mathsf{Combine}\left(d, r, (\hat{f}'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}\right),$$

where $\hat{f}'_{i,j}$ and $\mathcal{S}_{i,j}$ are defined as in Item 2a.

3. **Verifier decision phase:**

(a) **Poly-IOP decision:** Check that $\mathbf{V}_{\mathsf{poly}}^{\mathbb{y}, (\hat{f}_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{poly}}}) = 1$, where the k-th query to $\hat{f}_{i,j}$ is answered by $A_{i,j}[k]$ and queries to $\mathbb{y}$ are answered by querying $\mathbb{y}$ directly (reject if $\mathbf{V}_{\mathsf{poly}}$ rejects).
For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$, let $\mathcal{Q}_{i,j} \subseteq \mathbb{F}$ be the set of queries made by $\mathbf{V}_{\mathsf{poly}}$ to $\hat{f}_{i,j}$ and $\phi_{i,j} \colon [\mathsf{q}_i] \to \mathcal{Q}_{i,j}$ be the mapping where $\phi_{i,j}(x)$ returns the k-th query made to $\hat{f}_{i,j}$. Set $\mathcal{S}_i := \mathcal{Q}_{i,j} \cup \{x_i\}$.

(b) **Low-degree test decision:** Check that $\mathbf{V}_{\mathsf{prx}}$ accepts in its decision phase, when $\mathbf{V}$ answers a query t made by $\mathbf{V}_{\mathsf{prx}}$ to its input codeword $g \colon \mathcal{L} \to \mathbb{F}$ as follows.

i. For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$:

- Set $\mathsf{Ans}_{i,j} \colon \mathcal{S}_{i,j} \to \mathbb{F}$ and define the virtual function $\mathsf{Fill}_{i,j} \colon \mathcal{S}_{i,j} \to \mathbb{F}$ as follows:

$$\mathsf{Ans}_{i,j}(q) := \begin{cases} A_{i,j}[\phi_{i,j}^{-1}(q)] & q \in \mathcal{Q}_{i,j} \\ y_{i,j} & q = x_i \end{cases} \quad \mathsf{Fill}_{i,j}(q) := \begin{cases} w_{i,j}(\phi_{i,j}^{-1}(q)) & q \in \mathcal{Q}_{i,j} \\ w_{i,j}(\mathsf{q}_{i,j}+1) & q = x_i \end{cases}$$

- Define the virtual $f'_{i,j} := \mathsf{Quotient}(f_{i,j}, \mathcal{S}_{i,j}, \mathsf{Ans}_{i,j}, \mathsf{Fill}_{i,j})$.

ii. Define the virtual function

$$g := \mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}\right),$$

and answer according to this function by querying the functions $f_{i,j}$ or $w_{i,j}$ appropriately (observe that by the definition of $f'_{i,j}$ each query to it yields either a query to $f_{i,j}$ or to $w_{i,j}$ but not both).

(c) **Consistency with** Ans**:** For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$ query $f_{i,j}$ at every $x \in \mathcal{S}_{i,j} \cap \mathcal{L}$ and check that $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$.

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

- *Rounds.* The IOPP has $2 \cdot \mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}} + 1$ rounds. If $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ begins with a prover message, then the round complexity is reduced to $2 \cdot \mathsf{k}_{\mathsf{poly}} + \mathsf{k}_{\mathsf{prx}}$.

- *Proof length.* The oracle proof length (over the alphabet $\mathbb{F}$) is

$$\mathsf{l}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} |\mathcal{L}| + |S_{i,j}| \leq \mathsf{l}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} (|\mathcal{L}| + \mathsf{q}_{i,j} + 1) = \mathsf{l}_{\mathsf{prx}} + \mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}} \cdot (|\mathcal{L}| + 1)\,.$$

  The prover additionally sends $\sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} |S_{i,j}| \leq \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} \mathsf{q}_{i,j} + 1 = \mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}}$ field elements. Thus, the total proof length is

$$\mathsf{l}_{\mathsf{prx}} + 2 \cdot \mathsf{q}_{\mathsf{poly},\Pi} + s_{\mathsf{poly}} \cdot (|\mathcal{L}| + 2)\,.$$

- *Oracle input queries.* The verifier makes $\mathsf{q}_{\mathsf{poly},\mathbb{y}}$ queries to its input oracle $\mathbb{y}$.
- *Proof queries.* The verifier makes $\mathsf{q}_{\mathsf{prx},\Pi}$ queries to the internal messages of the proximity test, and $\mathsf{q}_{\mathsf{prx},f}$ queries to g. For every i, j, each query to g translates to a single query to either $f_{i,j}$ of $w_{i,j}$. Then, each $f_{i,j}$ is queried $\mathsf{q}_{i,j}$ times. Thus the total proof query complexity is $s_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\Pi} + \mathsf{q}_{\mathsf{poly},\Pi}$.
- *Verifier running time.* The verifier evaluates $\mathbf{V}_{\mathsf{poly}}$ in time $\mathsf{vt}_{\mathsf{poly}}$ and $\mathbf{V}_{\mathsf{prx}}$ in time $\mathsf{vt}_{\mathsf{prx}}$. Additionally, the verifier computes $\hat{\mathsf{Ans}}_{i,j}$ for every i, j, which requires time $O(\mathsf{q}_{i,j}^2)$.

  Therefore the verifier running time is $O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{vt}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \mathsf{q}_{i,j}^2)$.

5.6.2 Round-by-round knowledge soundness

We prove the round-by-round knowledge soundness of the IOPPP in Theorem 5.6.2.

Lemma 5.6.3. *Suppose that* $(\mathbf{P}_{\mathsf{poly}}, \mathbf{V}_{\mathsf{poly}})$ *has round-by-round knowledge soundness errors* $(\mathsf{err}_1^{\mathsf{poly}}, \ldots, \mathsf{err}_{\mathsf{k}_{\mathsf{poly}}}^{\mathsf{poly}})$*, and that* $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ *has round-by-round soundness errors* $(\mathsf{err}_1^{\mathsf{prx}}, \ldots, \mathsf{err}_{\mathsf{k}_{\mathsf{prx}}}^{\mathsf{prx}})$*. Suppose that for every* $i \in [\mathsf{k}_{\mathsf{poly}}]$ *and* $j \in [\ell_i]$*,* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i,j}]$ *is* $(\delta, \ell_{i,j})$*-list decodable. For every* $\delta \in (0, \min\{1 - \rho - 1/|\mathcal{L}|, 1 - \mathsf{B}(\rho)\})$*, the IOPP described in Theorem 5.6.2 has round-by-round knowledge soundness errors* $(\varepsilon_1^{\mathsf{out}}, \varepsilon_1^{\mathsf{piop}}, \ldots, \varepsilon_{\mathsf{k}_{\mathsf{poly}}}^{\mathsf{out}}, \varepsilon_{\mathsf{k}_{\mathsf{poly}}}^{\mathsf{piop}}, \varepsilon^{\mathsf{com}}, \varepsilon_1^{\mathsf{prx}}, \ldots, \varepsilon_{\mathsf{k}_{\mathsf{prx}}}^{\mathsf{prx}})$ *with extraction time* $O(\mathsf{et}_{\mathsf{poly}} + s_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}})$ *where:*

- $\varepsilon_i^{\mathsf{out}} \leq \frac{\sum_{j \in [\ell_i]} d_{i,j} \cdot \ell_{i,j}^2}{2 \cdot |\mathbb{F}|}$;
- $\varepsilon_i^{\mathsf{piop}} \leq \mathsf{err}_i^{\mathsf{poly}}(\mathbb{x}, \mathbb{y})$;
- $\varepsilon^{\mathsf{com}} \leq \mathsf{err}^{\star}\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{\ell_i} (d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right)$;
- $\varepsilon_i^{\mathsf{prx}} \leq \mathsf{err}_i^{\mathsf{prx}}(\delta)$.

Above, B *and* $\mathsf{err}^{\star}$ *are the proximity bound and error (respectively) described in Section 5.3.1.*

Proof. We describe the state function and prove the round-by-round knowledge soundness errors.

0. **State function for empty transcript.** Given an explicit input $\mathbb{x}$ and implicit input $\mathbb{y}$, we set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \emptyset) := \mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \emptyset)$.

1. **Bounding $\varepsilon_i^{\mathsf{out}}$.** At this stage, a partial transcript has the following form

$$\mathsf{tr} := (((f_{\ell,j})_{j\in[\ell_i]}, x_\ell, (y_{\ell,j})_{j\in[\ell_\ell]})_{\ell<i}, (f_{i,j})_{j\in[\ell_i]}) ,$$

and the verifier sends x_i.

- *State function.* We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||x_i) = 1$ if and only if at least one of the following holds.
  (a) There exist $\ell \le i$ and $j \in [\ell_i]$, and distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$ such that $\hat{u}_{\ell,j}(x_\ell) = \hat{u}'_{\ell,j}(x_\ell)$.
  (b) There exist codewords $(u_{\ell,j})_{\ell<i, j\in[\ell_\ell]}$ such that:
    i. $u_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$,
    ii. $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$, and
    iii. $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell<i}) = 1$.
- *Extractor.* The extractor $\mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})$ outputs $\perp$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$. We show that

$$\Pr_{x_i}[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||x_i) = 1] \le \varepsilon_i^{\mathsf{out}} = \frac{\sum_{j\in[\ell_i]} d_{i,j} \cdot \ell_{i,j}^2}{2 \cdot |\mathbb{F}|} .$$

Since the error is always below $\varepsilon_i^{\mathsf{out}}$, we do not need the extractor to be able to extract. We separate to two cases, and show that the state could only change since Item 1a holds for $\ell = i$:

– If $i = 1$ then according to the state function for the empty transcript described in Item 0, $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \emptyset) = \mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$, and so Item 1b does not hold. Moreover, since $i = 1$ this is the only choice of ℓ in Item 1a.
– If $i > 1$ then according to the state function defined in Item 2, there do not exist codewords $(u_{\ell,j})_{\ell<i, j\in[\ell_\ell]}$ such that $u_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$, $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ and,

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, (\hat{u}_{1,j})_{j\in[\ell_1]}, \alpha_1, \ldots, (\hat{u}_{i-1,j})_{j\in[\ell_{i-1}]}, \alpha_{i-1}) = 1 .$$

This is precisely stating that Item 1b does not hold. Moreover, by Item 2a, for every $\ell < i$, $j \in [\ell_i]$, and a pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$ it holds that $\hat{u}_{\ell,j}(x_\ell) \neq \hat{u}'_{\ell,j}(x_\ell)$. Therefore in order for Item 1a to hold, it must do so for $\ell = i$.

Since $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i,j}]$ is $(\delta, \ell_{i,j})$-list decodable, by Theorem 5.3.5 for every fixed $j \in [\ell_i]$ there exist a pair of distinct codewords $u_{i,j}, u'_{i,j} \in \Lambda(f_{i,j}, d_{i,j}, \delta)$ such that $\hat{u}_{i,j}(x_i) = \hat{u}'_{i,j}(x_i)$ with probability at most $\frac{d_{i,j}\cdot\ell_{i,j}^2}{2\cdot|\mathbb{F}|}$ over the choice of x_i. Applying the union bound over all choices of j, the probability that there exists a j for which this occurs is at most $\frac{\sum_{j\in[\ell_i]} d_{i,j}\cdot\ell_{i,j}^2}{2\cdot|\mathbb{F}|}$. As a result, Item 1a holds with probability at most $\frac{\sum_{j\in[\ell_i]} d_{i,j}\cdot\ell_{i,j}^2}{2\cdot|\mathbb{F}|}$.

2. **Bounding $\varepsilon_i^{\mathsf{piop}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := (((f_{\ell,j})_{j\in[\ell_\ell]}, x_\ell, (y_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell<i}, (f_{i,j})_{j\in[\ell_i]}, x_i, (y_{i,j})_{j\in[\ell_i]}),$$

and the verifier sends α_i.

- *State function.* We set $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 1$ if and only if at least one of the following hold:
  (a) There exist $\ell \le i$, $j \in [\ell_i]$, and a pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$ such that $\hat{u}_{\ell,j}(x_\ell) = \hat{u}'_{\ell,j}(x_\ell)$ or,
  (b) There exist codewords $(u_{\ell,j})_{\ell\le i, j\in[\ell_\ell]}$ such that:
    i. $u_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$,
    ii. $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ and,
    iii. $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell\le i}) = 1$.
- *Extractor.* The extractor $\mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})$ proceeds as follows:
  (a) For every $\ell < i$ and $j \in [\ell_\ell]$ compute $\Lambda(f_{\ell,j}, d_{\ell,j}, \delta) := \mathbf{E}_{\mathsf{RS}}(f_{\ell,j})$ and let $u_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$ be a codeword such that $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ (output $\perp$ if no such codeword exists).
  (b) Compute $\mathbb{w} := \mathbf{E}_{\mathsf{poly}}(\mathbb{y}, \mathbb{x}, ((\hat{u}_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell<i}, (\hat{u}_{i,j})_{j\in[\ell_i]})$ and output $\mathbb{w}$.

  The extractor runs in time at most $O(\mathsf{et}_{\mathsf{poly}} + \sum_{\ell<i} \ell_\ell \cdot \mathsf{et}_{\mathsf{RS}}) = O(\mathsf{et}_{\mathsf{poly}} + s_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}})$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{w}, \mathsf{tr}) = 0$ and that

$$\Pr_{\alpha_i}[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 1] > \varepsilon_i^{\mathsf{piop}} = \mathsf{err}_i^{\mathsf{poly}}(\mathbb{x}, \mathbb{y}).$$

  We show that $((\mathbb{x}, \mathbb{y}), \mathbf{E}(\mathbb{x}\mathbb{y}, \mathsf{tr})) \in \mathcal{R}$. Since $\mathsf{State}(\mathbb{x}, \mathbb{w}, \mathsf{tr}) = 0$, according to Item 1a, for every $\ell \le i$, $j \in [\ell_\ell]$ and pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$, it holds that $\hat{u}_{\ell,j}(x_\ell) \ne \hat{u}'_{\ell,j}(x_\ell)$. It follows that for every ℓ and j there exists at most one codeword $u_{\ell,j} \in \Lambda(f_{\ell,j}, d_{\ell,j}, \delta)$ with $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$. If for some j there is no such codeword, then, by definition, $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 0$ for every α_i, and so this cannot be the case.
  Suppose, then, that for every $\ell < i$ and $j \in [\ell_{i\ell}]$ there is a single such codeword $u_{\ell,j}$. This unique codeword will be found and chosen by $\mathbf{E}$. Since $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$, by Item 1b for every $(u_{\ell,j})_{\ell<i, j\in[\ell_i]}$ where $u_{\ell,j} \in \Lambda(f_\ell, d_{\ell,j}, \delta)$ and $\hat{u}_{\ell,j}(x_\ell) = y_{i,j}$, it holds that

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell<i}) = 0.$$

  Moreover, it holds that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 1$ only if

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[\ell_\ell]}, \alpha_\ell)_{\ell\le i}, (\hat{u}_{i,j})_{j\in[\ell_i]}, \alpha_i) = 1.$$

Thus, we get that

$$\begin{aligned}\Pr_{\alpha_i}\left[\mathsf{State}_{\mathsf{poly}}(\mathbb{x},\mathbb{y},((\hat{u}_{\ell,j})_{j\in[\ell_\ell]},\alpha_\ell)_{\ell<i},(\hat{u}_{i,j})_{j\in[\ell_i]},\alpha_i)=1\right] &= \Pr_{\alpha_i}\left[\mathsf{State}(\mathbb{x},\mathbb{y},\mathsf{tr}||\alpha_i)=1\right]\\ &> \varepsilon_i^{\mathsf{piop}}\\ &= \mathsf{err}_i^{\mathsf{poly}}(\mathbb{x},\mathbb{y})\,.\end{aligned}$$

It follows by knowledge soundness of the PIOP that

$$((\mathbb{x},\mathbb{y}),\mathbf{E}(\mathbb{x},\mathbb{y},\mathsf{tr}))=((\mathbb{x},\mathbb{y}),\mathbf{E}_{\mathsf{poly}}(\mathbb{x},\mathbb{y},((\hat{u}_{\ell,j})_{j\in[\ell_\ell]},\alpha_\ell)_{\ell\le i},(\hat{u}_{i,j})_{j\in[\ell_i]}))\in\mathcal{R}\,.$$

3. **Bounding $\varepsilon^{\mathsf{com}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr}:=(((f_{i,j})_{j\in[\ell_i]},x_i,(y_{i,j})_{j\in[\ell_\ell]},\alpha_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]},(A_{i,j},w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell_i]})\,,$$

and the verifier sends r. From $A_{i,j}$ we derive $\mathsf{Ans}_{i,j}$ and from $w_{i,j}$ we derive $\mathsf{Fill}_{i,j}$ as in Item 3b of the verifier decision algorithm.

- *State function.* We set $\mathsf{State}(f,\mathsf{tr}||r)=1$ if and only if all of the following hold:
  (a) $\mathbf{V}_{\mathsf{poly}}$ accepts given access to $\mathbb{w}$ and given query answers according to $A_{i,j}$ as in Item 2a of the verifier decision algorithm.
  (b) $\Delta(g,\mathsf{RS}[\mathbb{F},\mathcal{L},d])<\delta$.
  (c) For every $i\in[\mathsf{k}_{\mathsf{poly}}]$ and $j\in[\ell_i]$: $f_{i,j}(x)=\mathsf{Ans}_{i,j}(x)$ for every $x\in\mathcal{S}_{i,j}\cap\mathcal{L}$.
- *Extractor.* The extractor $\mathbf{E}(\mathbb{x},\mathbb{y},\mathsf{tr})$ always outputs $\perp$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x},\mathbb{w},\mathsf{tr})=0$. We show that

$$\Pr_r\left[\mathsf{State}(\mathbb{x},\mathbb{y},\mathsf{tr}||r)=1\right]\le\varepsilon^{\mathsf{com}}=\mathsf{err}^\star\left(d,\rho,\delta,\sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}}\ell_i\cdot(d-d_i+\mathsf{q}_i+2)\right)\,.$$

As the error is always below $\varepsilon^{\mathsf{com}}$, we do not need the extractor to be able to extract. Since $\mathsf{State}(\mathbb{x},\mathbb{w},\mathsf{tr})=0$, by Item 2 there is no set of codewords $(u_{\ell,j})_{\ell\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell_\ell]}$ for which all of the following hold:

(a) $u_{\ell,j}\in\Lambda(f_{\ell,j},d_\ell,\delta)$,
(b) $\hat{u}_{\ell,j}(x_\ell)=y_{\ell,j}$ and,
(c) $\mathsf{State}_{\mathsf{poly}}(\mathbb{x},\mathbb{y},((\hat{u}_{\ell,j})_{j\in[\ell_\ell]},\alpha_\ell)_{\ell\le\mathsf{k}_{\mathsf{poly}}})=1$.

In order for Item 3a to hold, the arrays $A_{i,j}$ must be such that $\mathbf{V}_{\mathsf{poly}}$ accepts given their values as oracle answers. Thus we can assume that the prover has sent such arrays. The following claim shows that there must be some i,j such that no codeword close to $f_{i,j}$ agrees with $\mathsf{Ans}_{i,j}$.

Claim 5.6.4. *There exists $i^*\in[\mathsf{k}_{\mathsf{poly}}]$ and $j^*\in[\ell_{i^*}]$ such that for every $u_{i^*,j^*}\in\Lambda(f_{i^*,j^*},d_{i^*,j^*},\delta)$ there exists $a\in\mathcal{S}_{i^*,j^*}$ such that $\hat{u}_{i^*,j^*}(a)\neq\mathsf{Ans}_{i^*,j^*}(a)$.*

Proof. Suppose towards contradiction that for every i, j there exists $u_{i,j} \in \Lambda(f_{i,j}, d_{i,j}, \delta)$ such that $\hat{u}_{i,j}(a) = \mathsf{Ans}_{i,j}(a)$ for every $a \in \mathcal{S}_{i,j}$ and fix some such set of codewords. Observe that $\hat{u}_{i,j}(a) = \mathsf{Ans}_{i,j}(a) = A_{i,j}[\phi_{i,j}(a)]$ for every $a \in \mathcal{Q}_{i,j}$. Since $\mathbf{V}_{\mathsf{poly}}$ accepts given query answers according to $A_{i,j}$ and since the $\hat{u}_{i,j}$ polynomials are consistent with $A_{i,j}$, it follows that

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{i,j})_{j\in[\ell_i]}, \alpha_i)_{i\le \mathsf{k}_{\mathsf{poly}}}) = 1\,.$$

This contradicts the assumption, coming from $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr})$ being equal 0, that the following cannot hold simultaneously:

(a) For every i, j: $u_{i,j} \in \Lambda(f_{i,j}, d_{i,j}, \delta)$,
(b) $\hat{u}_{\ell,j}(x_i) = y_{i,j} = \mathsf{Ans}_{i,j}(x_i)$ and,
(c) $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{i,j})_{j\in[\ell_i]}, \alpha_i)_{i\le \mathsf{k}_{\mathsf{poly}}}) = 1$.

□

Fix i^*, j^* as in Theorem 5.6.4. Together with Theorem 5.3.4, it follows that

$$\Delta(f'_{i^*,j^*}, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i^*,j^*} - |\mathcal{S}_{i^*,j^*}|]) + \frac{|\{x \in \mathcal{S}_{i^*,j^*} : f_{i^*,j^*}(x) \neq \mathsf{Ans}_{i^*,j^*}(x)\}|}{|\mathcal{L}|} > \delta\,.$$

Whether Item 3c holds is independent of the verifier randomness in this round, and so it must hold in order for the state function output to change to 1. Therefore $|\{x \in \mathcal{S}_{i^*,j^*} : f_{i^*,j^*}(x) \neq \mathsf{Ans}_{i^*,j^*}(x)\}| = 0$ and so

$$\Delta(f'_{i^*,j^*}, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i^*,j^*} - |\mathcal{S}_{i^*,j^*}|]) > \delta\,.$$

Therefore by recalling that

$$g := \mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}\right)\,,$$

and applying Theorem 5.3.13, which we can do since $\delta \in (0, \min\{1-\rho-1/|\mathcal{L}|, 1-\mathsf{B}(\rho)\})$:

$$\begin{aligned}
&\Pr_{r\leftarrow\mathbb{F}}[\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \le \delta] \\
&= \Pr_{r\leftarrow\mathbb{F}}\left[\Delta\left(\mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[\ell_i]}\right),\ \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]\right) \le \delta\right] \\
&> \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}}\sum_{j=1}^{\ell_i}(d - d_{i,j} + |\mathcal{S}_{i,j}| + 1)\right) \\
&\ge \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}}\sum_{j=1}^{\ell_i}(d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right)\,.
\end{aligned}$$

Finally,

4. **Bounding $\varepsilon_i^{\mathsf{prx}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := (((f_{i,j})_{j\in[\ell_i]}, x_i, (y_{i,j})_{j\in[\ell_\ell]}, \alpha_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}, (A_{i,j}, w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell_i]}, (\Pi_\ell, \alpha_{\mathsf{prx},\ell})_{\ell<i}, \Pi_i)\,,$$

where Π_ℓ and $\alpha_{\mathsf{prx},\ell}$ the ℓ-th prover and verifier message in the proximity test $(\mathbf{P}_{\mathsf{prx}}, \mathbf{V}_{\mathsf{prx}})$ respectively. The verifier sends proximity test message $\alpha_{\mathsf{prx},i}$.

- *State function.* We set $\mathsf{State}(f, \mathsf{tr}||\alpha_{\mathsf{prx},i}) = 1$ if and only if both of the following hold:
  (a) $\mathbf{V}_{\mathsf{poly}}$ accepts given access to $\mathbb{y}$ and given query answers according to $A_{i,j}$ as in Item 2a of the verifier decision algorithm.
  (b) Either $\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) < \delta$ or $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_i, \alpha_{\mathsf{prx},i}) = 1$.
  (c) For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$: $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$ for every $x \in \mathcal{S}_{i,j} \cap \mathcal{L}$.
- *Extractor.* The extractor $\mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})$ always outputs $\perp$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$. We show that

$$\Pr_{\alpha_{\mathsf{prx},i}}[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_{\mathsf{prx},i}) = 1] \leq \varepsilon_i^{\mathsf{prx}} = \mathsf{err}_i^{\mathsf{prx}}(\delta)\,.$$

  As the error is always below $\varepsilon_i^{\mathsf{prx}}$, we do not need the extractor to be able to extract. If the state is 0 this is due to the fact that there exists $i \in [\mathsf{k}_{\mathsf{poly}}]$, $j \in [\ell_i]$, and $x \in \mathcal{S}_{i,j} \cap \mathcal{L}$ such that $f_{i,j}(x) \neq \mathsf{Ans}_{i,j}(x)$ then Item 4c does not hold, in which case the state function must output 0. Suppose, then, that this is not the case. In order for the state to become to 1, it must be that Item 4a holds. We treat differently the cases of $i = 1$ and $i > 1$:
  – If $i = 1$ then, since Item 4a must hold, so does Item 3a. It follows that Item 3b does not hold, i.e., $\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \geq \delta$. By Item 4b, in order for the state to be 1 it must be that $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}) = 1$. Moreover, since $\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \geq \delta > 0$, we have that $\mathsf{State}_{\mathsf{prx}}(g, \emptyset) = 0$.
  – If $i > 1$ then both $\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) > \delta$ and $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_{i-1}, \alpha_{\mathsf{prx},i-1}) = 0$ by the assumption that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$. Since g cannot change, it must be that the state changes to 1, i.e., $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_i, \alpha_{\mathsf{prx},i}) = 1$

  In either cases, we have that:

$$\begin{aligned}
\varepsilon_i^{\mathsf{prx}} &= \Pr_{\alpha_{\mathsf{prx},i}}[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_{\mathsf{prx},1}) = 1 \mid \mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0] \\
&\leq \Pr_{\alpha_{\mathsf{prx},i}}[\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_i, \alpha_{\mathsf{prx},i}) = 1 \mid \mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_{i-1}, \alpha_{\mathsf{prx},i-1}) = 0] \\
&\leq \mathsf{err}_i^{\mathsf{prx}}(g) \\
&\leq \mathsf{err}_i^{\mathsf{prx}}(\delta)\,.
\end{aligned}$$

5. **Verifier decision.** We show that if $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$ for a full transcript tr, then the verifier rejects. The transcript has the form

$$\mathsf{tr} := (((f_{i,j})_{j\in[\ell_i]}, x_i, (y_{i,j})_{j\in[\ell_\ell]}, \alpha_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}, (A_{i,j}, w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[\ell_i]}, (\Pi_\ell, \alpha_{\mathsf{prx},\ell})_{\ell<\mathsf{k}_{\mathsf{prx}}},)\,,$$

If $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$ then, by Item 4 one of the following is true:

(a) $\mathbf{V}_{\mathsf{poly}}$ rejects given access to $\mathbb{y}$ and given query answers according to $A_{i,j}$ as in Item 2a of the verifier decision algorithm, or

(b) $\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \geq \delta$ and $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_{\mathsf{k}_{\mathsf{prx}}}, \alpha_{\mathsf{prx},\mathsf{k}_{\mathsf{prx}}}) = 0$.

(c) For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [\ell_i]$: $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$ for every $x \in \mathcal{S}_{i,j} \cap \mathcal{L}$.

The verifier in rejects if the first and third items do not hold, as this is tested directly. By round-by-round soundness of the proximity test, if $\mathsf{State}_{\mathsf{prx}}(g, \Pi_1, \alpha_{\mathsf{prx},1}, \ldots, \Pi_{\mathsf{k}_{\mathsf{prx}}}, \alpha_{\mathsf{prx},\mathsf{k}_{\mathsf{prx}}}) = 0$ then $\mathbf{V}_{\mathsf{prx}}$ rejects, and so if this is the case, then the verifier rejects. Thus the verifier must always reject if $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$.

□

5.7 Additional experimental data

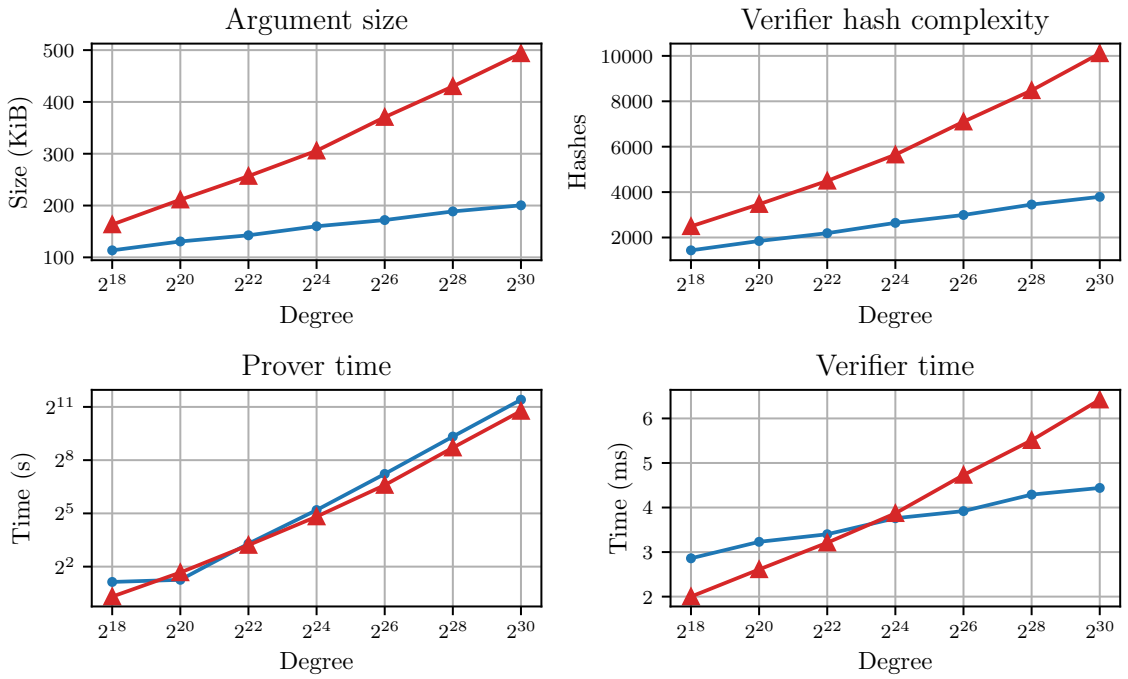
We collected additional experimental data, both in tabular and graphical forms. Figure 5.3 and Figure 5.4 further illustrate this information in graphical form, comparing STIR and FRI on each efficiency measure for various rates. Table 5.4 gives experimental results for the algebraic configuration. Finally, Table 5.5 compares the computed argument sizes of R1CS SNARGs based on STIR and FRI.

ρ \ d	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}
	Argument size (KiB, $\frac{\text{FRI}}{\text{STIR}}$)					
$1/2$	$\frac{207}{135} \approx 1.54\times$	$\frac{262}{159} \approx 1.64\times$	$\frac{333}{176} \approx 1.89\times$	$\frac{401}{201} \approx 1.99\times$	$\frac{478}{218} \approx 2.19\times$	$\frac{562}{242} \approx 2.32\times$
$1/4$	$\frac{126}{90} \approx 1.4\times$	$\frac{162}{107} \approx 1.51\times$	$\frac{199}{119} \approx 1.68\times$	$\frac{232}{136} \approx 1.7\times$	$\frac{274}{146} \approx 1.88\times$	$\frac{326}{165} \approx 1.97\times$
$1/8$	$\frac{97}{71} \approx 1.37\times$	$\frac{125}{87} \approx 1.43\times$	$\frac{151}{93} \approx 1.61\times$	$\frac{172}{111} \approx 1.55\times$	$\frac{206}{120} \approx 1.72\times$	$\frac{244}{135} \approx 1.81\times$
	Verifier time (ms, $\frac{\text{FRI}}{\text{STIR}}$)					
$1/2$	$\frac{439.4}{271.3} \approx 1.62\times$	$\frac{580.2}{343.4} \approx 1.69\times$	$\frac{764.2}{392.2} \approx 1.95\times$	$\frac{952.8}{470.2} \approx 2.03\times$	$\frac{1155.2}{519.4} \approx 2.22\times$	$\frac{1397.0}{596.6} \approx 2.34\times$
$1/4$	$\frac{283.0}{190.8} \approx 1.48\times$	$\frac{372.9}{239.5} \approx 1.56\times$	$\frac{470.8}{272.8} \approx 1.73\times$	$\frac{564.8}{331.6} \approx 1.7\times$	$\frac{684.5}{353.5} \approx 1.94\times$	$\frac{825.2}{410.2} \approx 2.01\times$
$1/8$	$\frac{227.3}{153.2} \approx 1.48\times$	$\frac{300.6}{200.3} \approx 1.5\times$	$\frac{366.6}{217.0} \approx 1.69\times$	$\frac{432.1}{269.6} \approx 1.6\times$	$\frac{527.9}{296.0} \approx 1.78\times$	$\frac{628.8}{340.5} \approx 1.85\times$
	Verifier hashes ($\frac{\text{FRI}}{\text{STIR}}$)					
$1/2$	$\frac{2562}{1409} \approx 1.82\times$	$\frac{3427}{1842} \approx 1.86\times$	$\frac{4547}{2171} \approx 2.09\times$	$\frac{5718}{2647} \approx 2.16\times$	$\frac{7005}{2971} \approx 2.36\times$	$\frac{8483}{3445} \approx 2.46\times$
$1/4$	$\frac{1680}{1028} \approx 1.63\times$	$\frac{2223}{1318} \approx 1.69\times$	$\frac{2841}{1536} \approx 1.85\times$	$\frac{3445}{1860} \approx 1.85\times$	$\frac{4171}{2039} \approx 2.05\times$	$\frac{5069}{2402} \approx 2.11\times$
$1/8$	$\frac{1369}{841} \approx 1.63\times$	$\frac{1815}{1120} \approx 1.62\times$	$\frac{2232}{1233} \approx 1.81\times$	$\frac{2647}{1554} \approx 1.7\times$	$\frac{3237}{1722} \approx 1.88\times$	$\frac{3877}{1997} \approx 1.94\times$

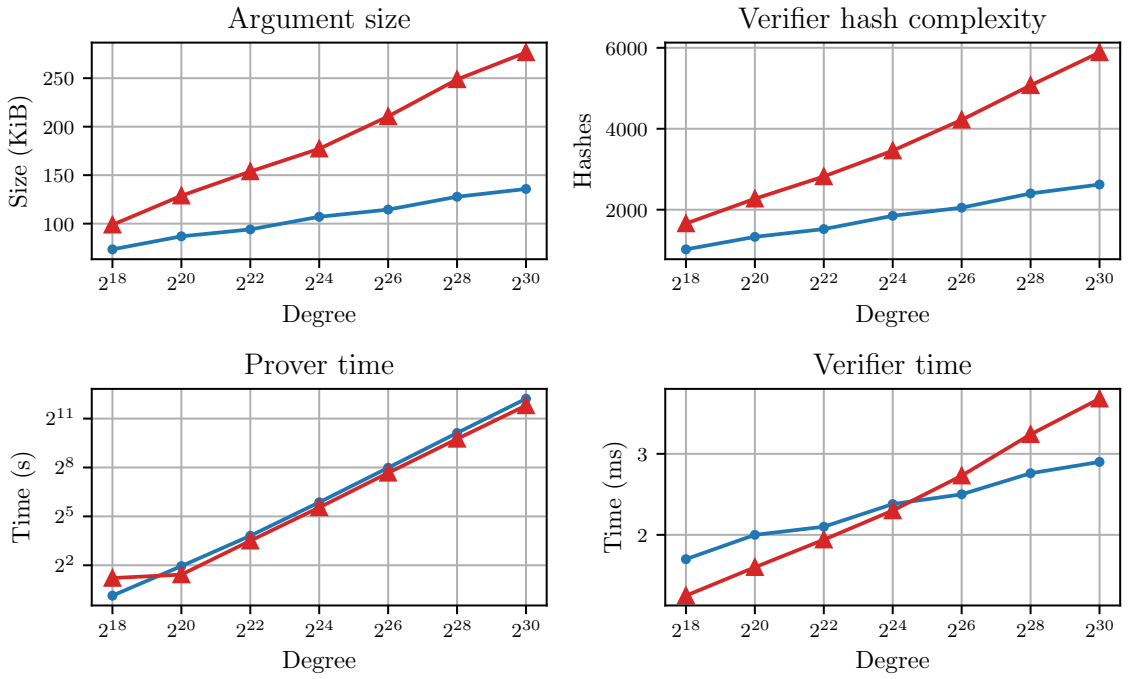
Table 5.4: Comparison of concrete costs between STIR and FRI when using Poseidon. The numerator of the fraction is the cost associated to FRI, while the denominator is that associated to STIR. For all metrics, lower is better.

ρ \ n	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
	Argument size (KiB, $\frac{\text{FRI}}{\text{STIR}}$)						
$1/2$	$\frac{253}{155} \approx 1.63\times$	$\frac{312}{178} \approx 1.75\times$	$\frac{353}{196} \approx 1.8\times$	$\frac{422}{220} \approx 1.92\times$	$\frac{494}{238} \approx 2.08\times$	$\frac{548}{262} \approx 2.09\times$	$\frac{631}{280} \approx 2.25\times$
$1/4$	$\frac{145}{102} \approx 1.42\times$	$\frac{176}{117} \approx 1.5\times$	$\frac{201}{128} \approx 1.57\times$	$\frac{236}{144} \approx 1.64\times$	$\frac{274}{155} \approx 1.77\times$	$\frac{305}{171} \approx 1.78\times$	$\frac{347}{182} \approx 1.91\times$
$1/8$	$\frac{106}{79} \approx 1.34\times$	$\frac{128}{92} \approx 1.39\times$	$\frac{145}{100} \approx 1.45\times$	$\frac{170}{114} \approx 1.49\times$	$\frac{197}{122} \approx 1.61\times$	$\frac{218}{136} \approx 1.6\times$	$\frac{248}{144} \approx 1.72\times$
$1/16$	$\frac{90}{70} \approx 1.29\times$	$\frac{108}{82} \approx 1.32\times$	$\frac{121}{88} \approx 1.38\times$	$\frac{141}{100} \approx 1.41\times$	$\frac{163}{108} \approx 1.51\times$	$\frac{179}{120} \approx 1.49\times$	$\frac{203}{127} \approx 1.6\times$

Table 5.5: Comparison of argument size of a SNARK for R1CS using STIR or FRI as their low-degree test. The numerator of the fraction is the cost associated to FRI, while the denominator is that associated to STIR. Lower is better.

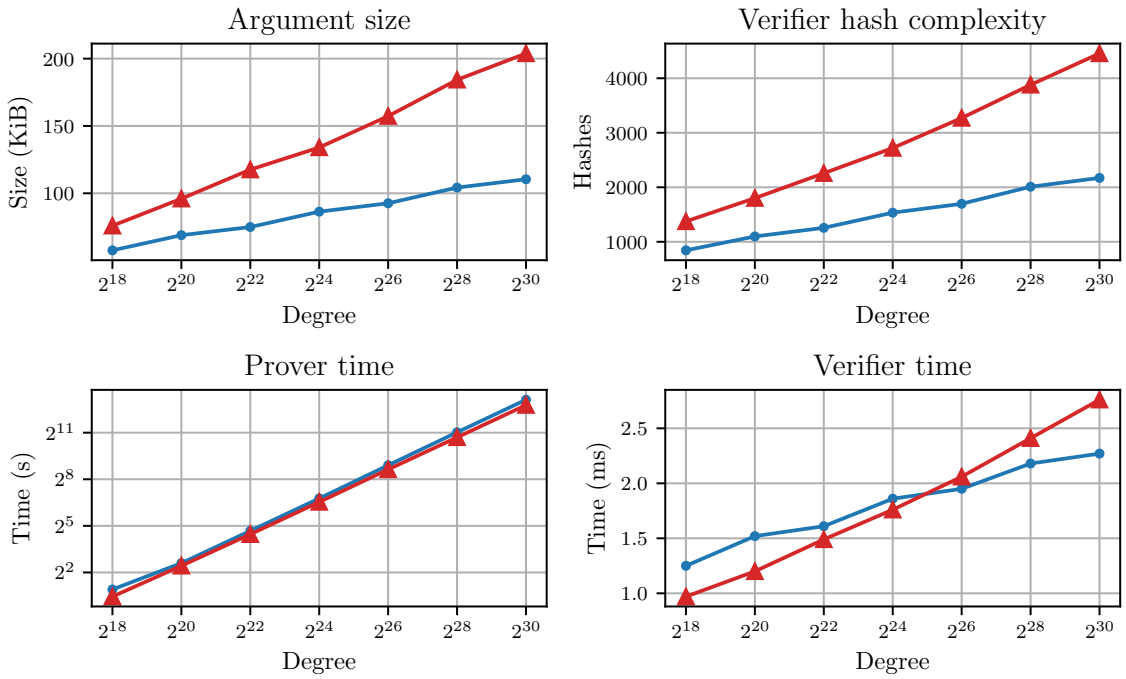


(a) $\rho = 1/2$, FRI: ▲, STIR: ●.

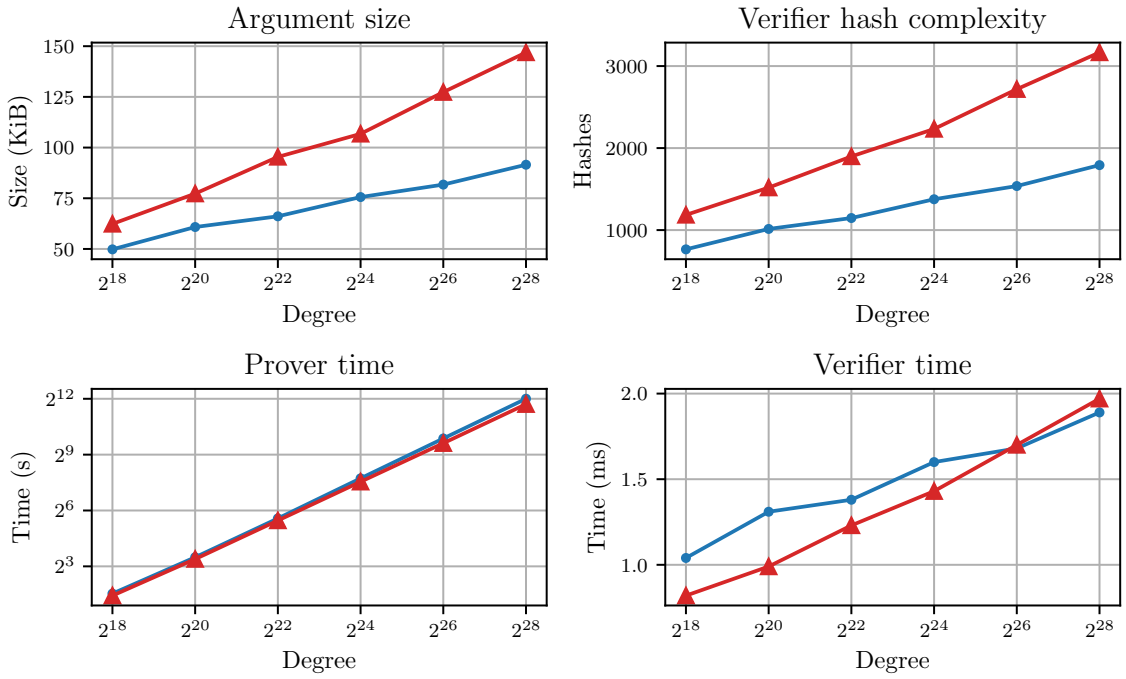


(b) $\rho = 1/4$, FRI: ▲, STIR: ●.

Figure 5.3: Comparison of FRI and STIR. Figure 5.3a is for $\rho = 1/2$, Figure 5.3b for $\rho = 1/4$. Lower is better.



(a) $\rho = 1/8$, FRI: ▲, STIR: ●.



(b) $\rho = 1/16$, FRI: ▲, STIR: ●.

Figure 5.4: Comparison of FRI and STIR. Figure 5.4a is for $\rho = 1/8$, Figure 5.4b for $\rho = 1/16$. Lower is better.

5.8 A poly-IOP for R1CS

We give a polynomial IOP for the R1CS relation.

Definition 5.8.1. *The relation $\mathcal{R}_{\mathsf{R1CS}}$ is the set of all pairs $((\mathbb{F}, k, n, A, B, C, v), w)$ where $\mathbb{F}$ is a finite field, $k, n \in \mathbb{N}$ (with $k \leq n$), A, B, C are $n \times n$ matrices over $\mathbb{F}$, $v \in \mathbb{F}^k$, and $w \in \mathbb{F}^{n-k}$, such that for all $i \in [n]$:*

$$\left(\sum_{j=0}^{n} A_{i,j} \cdot z_j\right) \cdot \left(\sum_{j=0}^{n} B_{i,j} \cdot z_j\right) = \sum_{j=0}^{n} C_{i,j} \cdot z_j \,,$$

where $z := (v, w) \in \mathbb{F}^n$.

Theorem 5.8.2. *There is an IOP for $\mathcal{R}_{\mathsf{R1CS}}$ with the following properties.*
- Round complexity: *2.*
- Number of polynomials: 5 *with degree at most n,* 1 *with degree at most $n - k$, and* 1 *with degree at most $n - 1$.*
- Query complexity: *7.*
- Round-by-round knowledge soundness error: $(\varepsilon^{\mathsf{shift}}, \varepsilon^{\mathsf{dec}})$ *with extraction time $O(n \cdot (n - k))$, where $\varepsilon^{\mathsf{shift}} \leq \frac{3n}{|\mathbb{F}|}$ and $\varepsilon^{\mathsf{dec}} \leq \frac{2n}{|\mathbb{F}|}$.*

5.8.1 Construction

Construction 5.8.3. Let $\mathbb{F}$ be a field. Consider the following ingredients and notation:

- H is a subgroup of $\mathbb{F}^*$ of order n. We sometimes refer to elements of H as elements in $[|H|]$. Implicitly, we assume a bijection between the two and use it as appropriate to translate between the two domains. Thus, for $S \subseteq H$ we refer to $f \colon S \to \mathbb{F}$ and $f \in \mathbb{F}^{|S|}$ interchangeably.
- $H_{\mathsf{in}} \subseteq H$ is the subset of order $|H_{\mathsf{in}}| = k$ that corresponds to the indices $\{1, \ldots, k\}$.
- $\hat{V}_H \in \mathbb{F}^{<n+1}[X]$ and $\hat{V}_{H_{\mathsf{in}}} \in \mathbb{F}^{<k+1}[X]$ are the unique non-zero polynomials that are 0 on H and H_{in}.
- For $r \in \mathbb{F}$, $\hat{p}_r \in \mathbb{F}^{<n}[X]$ is the unique polynomial such that $\hat{p}_r(x) := r^x$ for every $x \in H$.
- For matrix M and $r \in \mathbb{F}$, $\hat{q}_{M,r} \in \mathbb{F}^{<n}[X]$ is the unique univariate polynomial which satisfies: $\hat{q}_{M,r}(x) := \sum_{b \in H} r^b \cdot M^\top(x, b)$ for every $x \in H$.

The IOP proceeds as follows.

- **Inputs:** The honest prover is given $((\mathbb{F}, k, n, A, B, C, v), w) \in \mathcal{R}_{\mathsf{R1CS}}$, and the verifier is given $(\mathbb{F}, k, n, A, B, C, v)$.
- **Interaction phase:**
  1. **Commit to witness polynomials:** The prover sends polynomials $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{f}_0 \in \mathbb{F}^{<n}[X]$ and $\hat{f}_w \in \mathbb{F}^{<n-k}[X]$. In the honest case, the prover sets these polynomials as follows:

(a) Let $z := (v, w) \in \mathbb{F}^n$. 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) $\hat{f}_0(X) := \frac{\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X)}{\hat{V}_H(X)}$.
(c) $\hat{f}_w(X) := \frac{\hat{z}(X) - \hat{v}(X)}{\hat{V}_{H_{\mathsf{in}}(X)}}$ where $\hat{z} \in \mathbb{F}^{<n}[X]$ is the unique low-degree polynomial that is equal to z on H and $\hat{v} \in \mathbb{F}^{<k}[X]$ is the unique low-degree polynomial that is equal to v on H_{in}.

2. **Randomize polynomials:** The verifier sends $r \leftarrow \mathbb{F}$.
3. **Univariate sumcheck proof:** The prover sends polynomials $\hat{g}_1 \in \mathbb{F}^{<n}[X]$ and $\hat{g}_2 \in \mathbb{F}^{<n-1}[X]$. In the honest case, the prover defines the following:
(a) $\hat{f}_z(X) := \hat{f}_w(X) \cdot \hat{V}_{H_{\mathsf{in}}}(X) + \hat{v}(X)$.
(b)

$$\begin{aligned}\hat{u}(X) := \hat{p}_r(X) \cdot \hat{f}_A(X) - \hat{q}_{A,r}(X) \cdot \hat{f}_z(X) \\ + r^n \cdot \left(\hat{p}_r(X) \cdot \hat{f}_B(X) - \hat{q}_{B,r}(X) \cdot \hat{f}_z(X)\right) \\ + r^{2n} \cdot \left(\hat{p}_r(X) \cdot \hat{f}_C(X) - \hat{q}_{C,r}(X) \cdot \hat{f}_z(X)\right)\end{aligned}$$

Then $\hat{g}_1$ and $\hat{g}_2$ are the unique polynomials such that $\hat{u}(X) := \hat{V}_H(X) \cdot \hat{g}_1(X) + X \cdot \hat{g}_2(X)$.

- **Verifier decision phase:** Sample $\alpha \leftarrow \mathbb{F}$ and query $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{f}_0, \hat{f}_w, \hat{g}_1, \hat{g}_2$ each at α. Accept if the following checks pass.

1. **Zero test:** $\hat{f}_A(\alpha) \cdot \hat{f}_B(\alpha) - \hat{f}_C(\alpha) = \hat{f}_0(\alpha) \cdot \hat{V}_H(\alpha)$.
2. **Univariate sumcheck test:** $\hat{u}(\alpha) = \hat{g}_1(\alpha) \cdot \hat{V}_H(\alpha) + a \cdot \hat{g}_2(\alpha)$. Evaluating $\hat{u}(\alpha)$ on reduces to computing:
(a) $\hat{f}_z(\alpha) := \hat{f}_w(\alpha) \cdot \hat{V}_{H_{\mathsf{in}}}(\alpha) + \hat{v}(\alpha)$.
(b)

$$\begin{aligned}\hat{u}(\alpha) := \hat{p}_r(\alpha) \cdot \hat{f}_A(\alpha) - \hat{q}_{A,r}(\alpha) \cdot \hat{f}_z(\alpha) \\ + r^n \cdot \left(\hat{p}_r(\alpha) \cdot \hat{f}_B(\alpha) - \hat{q}_{B,r}(\alpha) \cdot \hat{f}_z(\alpha)\right) \\ + r^{2n} \cdot \left(\hat{p}_r(\alpha) \cdot \hat{f}_C(\alpha) - \hat{q}_{C,r}(\alpha) \cdot \hat{f}_z(\alpha)\right)\end{aligned}$$

(The verifier can compute $\hat{V}_{H_{\mathsf{in}}}(\alpha)$, $\hat{v}(\alpha)$, $\hat{p}_r(\alpha)$, $\hat{q}_{A,r}(\alpha)$, $\hat{q}_{B,r}(\alpha)$, and $\hat{q}_{C,r}(\alpha)$ by itself.)

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

- *Rounds.* The IOPP has 2 rounds.

- *Number of polynomials.* The prover sends 5 polynomials with degree less than n in the first round, and 2 polynomials in the second round, one of which has degree less than n and the second has degree less than $n-1$.
- *Proof queries.* The verifier makes 7 queries in total, each to a different polynomial, and all at the same point.

Preliminaries. In the proof of completeness and soundness we use the following fact, showing another description of one of the polynomials described in the protocol:

Fact 5.8.4. *For every $r \in \mathbb{F}$:*

$$\sum_{a\in H} \hat{p}_r(a)\cdot \hat{f}_M(a) - \hat{q}_{M,r}(a)\cdot \hat{f}(a) = \sum_{a\in H}\left(\hat{f}_M(a) - \sum_{b\in H} M(a,b)\cdot \hat{f}_z(b)\right)\cdot r^a\,.$$

Proof. The fact follows by opening up the expressions:

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

where in the second equality we have renamed the variables and switched the order of the sums. □

We use the following lemma, implementing a univariate sumcheck, first shown in [BCRSVW19], and described in the form below in [ACY23]:

Lemma 5.8.5 ([BCRSVW19])**.** *Let H be a multiplicative subgroup of $\mathbb{F}^*$. Let $\hat{f} \in \mathbb{F}^{<d}[X]$ be a polynomial, and $\gamma \in \mathbb{F}$ be a claimed sum. Then:*

- ***Completeness.*** *If $\sum_{a\in H}\hat{f}(a) = \gamma$ then*

$$\Pr_{\alpha\leftarrow\mathbb{F}}\left[\begin{array}{l|l}\hat{g}_1 \in \mathbb{F}^{<|H|-1}[X] & \hat{f}(X) \equiv \\ \wedge\ \hat{g}_2 \in \mathbb{F}^{<d-|H|+1}[X] & \hat{g}_1(X)\cdot \hat{V}_H(X) + (X\cdot \hat{g}_2(X) + \frac{\gamma}{|H|}) \\ \wedge\ \hat{f}(\alpha) = \hat{g}_1(\alpha)\cdot \hat{V}_H(\alpha) + (\alpha\cdot \hat{g}_2(\alpha) + \gamma/|H|) & \end{array}\right] = 1\,.$$

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

$$\Pr_{\alpha\leftarrow\mathbb{F}}\left[\begin{array}{l|l}\hat{g}_1 \in \mathbb{F}^{<|H|-1}[X] & \\ \wedge\ \hat{g}_2 \in \mathbb{F}^{<d-|H|+1}[X] & (\hat{g}_1,\hat{g}_2) \leftarrow \tilde{\mathbf{P}} \\ \wedge\ \hat{f}(\alpha) = \hat{g}_1(\alpha)\cdot \hat{V}_H(\alpha) + (\alpha\cdot \hat{g}_2(\alpha) + \gamma/|H|) & \end{array}\right] \leq \frac{d}{|\mathbb{F}|}\,.$$

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* $O(\log|H|)$ *(field operations).*

5.8.2 Completeness

Consider an instance $\mathbb{x} := (\mathbb{F}, k, n, A, B, C, v)$ and corresponding witness $\mathbb{w} = w$ with $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{R1CS}}$. First note that, for every $a \in H$,

$$\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = (Az)(a) \cdot (Bz)(a) - (Cz)(a) = 0\,,$$

and hence $\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X)$ is divisible by $\hat{V}_H$. Thus, Item 1 will always succeed, as both sides are identical as polynomials. Next, by Theorem 5.8.4, for $M \in \{A, B, C\}$ and $r \in \mathbb{F}$ we have that

$$\sum_{a \in H} \hat{p}_r(a) \cdot \hat{f}_M(a) - \hat{q}_{M,r}(a) \cdot \hat{f}(a) = \sum_{a \in H} \left(\hat{f}_M(a) - \sum_{b \in H} M(a,b) \cdot \hat{f}_z(b) \right) \cdot r^a = 0\,,$$

where the last equality follows since, for every $a \in H$, we have $\hat{f}_M(a) = (Mz)(a) = \sum_{b \in H} M(a,b) \cdot z(b)$. Thus $\sum_{a \in H} \hat{u}(a) = 0$ and by Theorem 5.8.5 the check in Item 2 succeeds with probability 1.

5.8.3 Round-by-round knowledge soundness

Lemma 5.8.6. *The poly-IOP in Theorem 5.8.3 has round-by-round knowledge soundness errors* $(\varepsilon^{\mathsf{shift}}, \varepsilon^{\mathsf{dec}})$ *with extraction time* $O(n \cdot (n-k))$ *where:*

- $\varepsilon^{\mathsf{shift}} \leq \frac{3n}{|\mathbb{F}|}$.
- $\varepsilon^{\mathsf{dec}} \leq \frac{2n}{|\mathbb{F}|}$.

State function and proof. We define the state function, and prove bounds on the round-by-round soundness error.

0. **State function for empty transcript.** Given an input $\mathbb{x} := (\mathbb{F}, k, n, A, B, C, v)$ we set $\mathsf{State}(\mathbb{x}, \emptyset) = 1$ if and only if $\mathbb{x} \in L(\mathcal{R}_{\mathsf{R1CS}})$.

1. **Bounding** $\varepsilon^{\mathsf{shift}}$**.** At this stage, the partial transcript has the form $\mathsf{tr} := (\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{f}_0, \hat{f}_w)$ and the verifier sends r.

   - *State function.* We set $\mathsf{State}(\mathbb{x}, \mathsf{tr}||r) = 1$ if and only if both of the following hold:
     (a) $\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X) \equiv \hat{f}_0(X) \cdot \hat{V}_H(X)$ and
     (b) $\sum_{a \in H} \hat{u}(a) = 0$, where $\hat{u}$ is defined as in Item 2 of the verifier's decision algorithm.

- *Extractor.* Given $\mathbb{x}$ and tr, the extractor computes $\hat{z}(X) := \hat{f}_w(X) \cdot \hat{V}_{H_{\mathsf{in}}}(X) + \hat{v}(X)$ and then outputs $w \colon H \setminus H_{\mathsf{in}} \to \mathbb{F}$, where $w(i) = \hat{z}(i)$. The extractor runs in time $O(n \cdot (n-k))$ given the coefficients of $\hat{f}_w$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathbb{w}, \mathsf{tr}) = 0$ and that

$$\Pr_r \left[\mathsf{State}(\mathbb{x}, \mathsf{tr}||r) = 1\right] > \varepsilon^{\mathsf{shift}} = \frac{3n}{|\mathbb{F}|}.$$

We show that $(\mathbb{x}, \mathbf{E}(\mathbb{x}, \mathsf{tr})) \in \mathcal{R}_{\mathsf{R1CS}}$. Define $\hat{h} \in \mathbb{F}[X]$ as follows:

$$\begin{aligned}\hat{h}(X) := &\sum_{a \in H} \left(\hat{f}_A(a) - \sum_{b \in H} A(a,b) \cdot \hat{f}_z(b) \right) \cdot X^a \\ &+ \sum_{a \in H} \left(\hat{f}_B(a) - \sum_{b \in H} B(a,b) \cdot \hat{f}_z(b) \right) \cdot X^{n+a} \\ &+ \sum_{a \in H} \left(\hat{f}_C(a) - \sum_{b \in H} C(a,b) \cdot \hat{f}_z(b) \right) \cdot X^{2n+a}.\end{aligned}$$

Observe that $\deg(\hat{h}) \leq 2n + |H| = 3n$, and that for r chosen by the verifier, by Theorem 5.8.4 the evaluation $\hat{h}(r)$ is equivalent to $\sum_{a \in H} \hat{u}(a)$. By the polynomial identity lemma, since:

$$\Pr_{r \leftarrow \mathbb{F}} \left[\hat{h}(r) = 0\right] = \Pr_{r \leftarrow \mathbb{F}} \left[\sum_{a \in H} \hat{u}(a) = 0\right] \geq \Pr_{r \leftarrow \mathbb{F}} \left[\mathsf{State}(\mathbb{x}, \mathsf{tr}||r) = 1\right] > 3n,$$

it holds that $\hat{h}$ is the zero polynomial. Consequently, for every $M \in \{A, B, C\}$:

$$\hat{f}_M(X) = \sum_{b \in H} M(X,b) \cdot \hat{f}_z(b) = \sum_{b \in H} M(X,b) \cdot \left(\hat{f}_w(b) \cdot \hat{V}_{H_{\mathsf{in}}}(b) + \hat{v}(b) \right).$$

Letting $w := \mathbf{E}(\mathbb{x}, \mathsf{tr})$, and $z \colon H \to \mathbb{F}$ where $z(i) = w(i)$ for $i \in H \setminus H_{\mathsf{in}}$ and $z(i) = v(i)$ otherwise (while $v \in \mathbb{F}^n$, recall that we have a one-to-one correspondence between H and $[|H|]$, so use this to we map v to a function $\mathbb{F}^{H_{\mathsf{in}}}$), for every $M \in \{A, B, C\}$ and $a \in H$ it holds that

$$\hat{f}_M(a) = \sum_{b \in H} M(a,b) \cdot z(b).$$

In order for the prover to have $\mathsf{State}(\mathbb{x}, \mathsf{tr}||r) = 1$, by Item 1a, it must be that $\hat{f}_A(X) \cdot \hat{f}_B(X) - \hat{f}_C(X) \equiv \hat{f}_0(X) \cdot \hat{V}_H(X)$, which implies that for every $a \in H$,

$\hat{f}_A(a) \cdot \hat{f}_B(a) = \hat{f}_C(a)$. Therefore, for every $a \in H$,

$$\begin{aligned}\left(\sum_{b\in H} A(a,b)\cdot z(b)\right)\cdot\left(\sum_{b\in H} B(a,b)\cdot z(b)\right) &= \hat{f}_A(a)\cdot\hat{f}_B(a)\\ &= \hat{f}_C(a)\\ &= \left(\sum_{b\in H} C(a,b)\cdot z(b)\right).\end{aligned}$$

Consequently, since $z = (v, w)$, it holds that $(\mathbb{x}, w) \in \mathcal{R}_{\mathsf{R1CS}}$.

2. **Bounding $\varepsilon^{\mathsf{dec}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := ((\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{h}, \hat{f}_w), r, (\hat{g_1}, \hat{g}_2)),$$

and the verifier sends α.

- *State function.* We set $\mathsf{State}(\mathbb{x}, \mathsf{tr}||\alpha) = 1$ if and only if both of the following hold:
  (a) $\hat{f}_A(\alpha)\cdot\hat{f}_B(\alpha) - \hat{f}_C(\alpha) = \hat{f}_0(\alpha)\cdot\hat{V}_H(\alpha)$ and
  (b) $\hat{u}(\alpha) = \hat{g}_1(\alpha)\cdot\hat{V}_H(\alpha) + \alpha\cdot\hat{g}_2(\alpha)$, where $\hat{u}$ is defined as in Item 2 of the verifier's decision algorithm.

  (Observe that this is what the verifier checks, and so if $\mathsf{State}(\mathbb{x}, \mathsf{tr}||\alpha) = 0$, then the verifier rejects.)
- *Extractor.* Given $\mathbb{x}$ and tr, the extractor outputs $\perp$.
- *Bounding the error.* Suppose that $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$. We show that

$$\Pr_{\alpha}\left[\mathsf{State}(\mathbb{x}, \mathsf{tr}||\alpha) = 1\right] \leq \varepsilon^{\mathsf{dec}} = \frac{2n}{|\mathbb{F}|}.$$

  Since the error is always below $\varepsilon^{\mathsf{dec}}$, we do not need the extractor to be able to extract. Since $\mathsf{State}(\mathbb{x}, \mathsf{tr}) = 0$, by Item 1, one of the following is true:
  - $\hat{f}_A(X)\cdot\hat{f}_B(X) - \hat{f}_C(X) \neq \hat{f}_0(X)\cdot\hat{V}_H(X)$. If this is the case for $\alpha \leftarrow \mathbb{F}$ it holds that $\hat{f}_A(\alpha)\cdot\hat{f}_B(\alpha) - \hat{f}_C(\alpha) = \hat{f}_0(\alpha)\cdot\hat{V}_H(\alpha)$ with probability at most $2n/|\mathbb{F}|$ (note that $\hat{f}_A(X)\cdot\hat{f}_B(X) - \hat{f}_C(X) = \hat{f}_0(X)\cdot\hat{V}_H(X)$ has degree bounded by $2n$). Thus in this case, by Item 2a, $\mathsf{State}(\mathbb{x}, \mathsf{tr}||\alpha) = 1$ with probability at most $2n/|\mathbb{F}|$.
  - If $\sum_{a\in H}\hat{u}(a) \neq 0$, then by Theorem 5.8.5, for every $\hat{g}_1 \in \mathbb{F}^{<n}[X]$ and $\hat{g}_2 \in \mathbb{F}^{<n-1}[X]$, the probability over the choice of $\alpha \leftarrow \mathbb{F}$ that $\hat{u}(\alpha) = \hat{g}_1(\alpha)\cdot\hat{V}_H(\alpha) + \alpha\cdot\hat{g}_2(\alpha)$ is at most $2n/|\mathbb{F}|$. Thus in this case, by Item 2b, $\mathsf{State}(\mathbb{x}, \mathsf{tr}||\alpha) = 1$ with probability at most $2n/|\mathbb{F}|$.

  Taking both cases into consideration, we have that $\mathsf{State}(\mathbb{x}, \mathsf{tr}||a) = 1$ with probability at most $2n/|\mathbb{F}|$.

5.9 Derivations for Section 5.4.3

We derive bounds on the parameters of the IOPPs described in Section 5.4.3.

- In Section 5.9.1 we give derivations for computing provable security bounds.
- In Section 5.9.2 we give derivations for computing security bounds assuming Theorem 5.4.6 with $c_1 = c_2 = c_3 = 1$.

The derivations in both sections use the following bound about a variant of the geometric sum:

Fact 5.9.1. $\sum_{i=1}^{M} \frac{1}{i+c} < \log\left(\frac{M}{c} + 1\right) + 1$ *for every* $c > 0$.

Proof. Let $s := \lfloor c \rfloor$ be the nearest integer smaller than c, and let H_m be the m-th harmonic number. Recall that $\ln(m+1) \leq H_m \leq \ln(m) + 1$. Then

$$\begin{aligned}
\sum_{i=1}^{M} \frac{1}{i+c} &\leq \sum_{i=1}^{M} \frac{1}{i+s} \\
&= H_{M+s} - H_s \\
&\leq \ln(M+s) + 1 - \ln(s+1) \\
&\leq \ln(M+c) + 1 - \ln(c) \\
&= \ln\left(\frac{M}{c} + 1\right) + 1 \\
&< \log\left(\frac{M}{c} + 1\right) + 1 .
\end{aligned}$$

where the final inequality holds since $\log(x) > \ln(x)$ for $x > 1$. □

We additionally bound the rate of the iterations from below:

Fact 5.9.2. $\rho_i \geq \rho / d$.

Proof. $\rho_i = (2/k)^i \cdot \rho \geq \rho / k^M = \rho / k^{\lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor} \geq \rho / d$. □

5.9.1 Provable security

We bound the properties of the IOPP for provable soundness error described in Section 5.4.3. We begin by showing that $\eta_i < \sqrt{\rho_i}/20$, then bound the complexity parameters, and finally bound round-by-round soundness error of the protocol. Note that these parameters are based on the improved ones derived in Theorem 5.4.3.

Bounds on η values. We show that since

$$|\mathbb{F}| > 10^7 \cdot (\lambda+1) \cdot 2^{\lambda+1} \cdot d^2 \cdot |\mathcal{L}|^{3.5} \cdot \left(1 + \max\left\{\left\lceil \frac{1}{-\log(1-\delta')} \right\rceil, \left\lceil \frac{1}{-\log(1.05 \cdot \sqrt{\rho})} \right\rceil\right\}\right),$$

it holds that $\eta_i \leq \sqrt{\rho_i}/20$ for every i. First observe that

$$|\mathbb{F}| > 10^7 \cdot 2^{\lambda+1} \cdot d^2 \cdot |\mathcal{L}|^{3.5} \cdot (1 + \max\{t_0, t_i\}) ,$$

We now bound the η parameters.

- $i = 0$:

$$\eta_0 = \left(\frac{2^\lambda \cdot (k-1) \cdot (d/k)^2}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} < \left(\frac{2^\lambda \cdot d^2}{2^7} \cdot \frac{1}{10^7 \cdot 2^\lambda \cdot d^2 \cdot |\mathcal{L}|^{3.5}}\right)^{1/7} = \frac{1}{20 \cdot \sqrt{|\mathcal{L}|}} < \frac{\sqrt{\rho}}{20} .$$

- $i > 0$:

$$\eta_i := \max\left\{\left(\frac{2^\lambda \cdot d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)}\right)^{1/2}, \left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2\right)}{2^7 \cdot |\mathbb{F}|}\right)^{1/7}\right\} .$$

We show that both options are bounded by $\frac{\sqrt{\rho/d}}{20} \leq \frac{\sqrt{\rho_i}}{20}$.

– First option:

$$\begin{aligned}\left(\frac{2^\lambda \cdot d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)}\right)^{1/2} &< \left(\frac{2^\lambda \cdot |\mathcal{L}|}{8 \cdot (|\mathbb{F}| - |\mathcal{L}|)}\right)^{1/2} \\ &\leq \left(\frac{2^\lambda \cdot |\mathcal{L}|}{8} \cdot \frac{1}{10^7 \cdot 2^\lambda \cdot d \cdot |\mathcal{L}|^2}\right)^{1/2} \\ &\leq \left(\frac{1}{20^2 \cdot d \cdot |\mathcal{L}|}\right)^{1/2} \\ &= \frac{\sqrt{\rho/d}}{20} .\end{aligned}$$

– Second option:

$$\begin{aligned}&\left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2\right)}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} \\ &< \left(\frac{2^{\lambda+1} \cdot (t_{i-1}+1) \cdot d^2}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} \\ &< \left(\frac{2^{\lambda+1} \cdot (t_{i-1}+1) \cdot d^2}{2^7} \cdot \frac{1}{10^7 \cdot 2^{\lambda+1} \cdot d^2 \cdot |\mathcal{L}|^{3.5} \cdot (t_{i-1}+1)}\right)^{1/7} \\ &< \left(\frac{1}{20^7 \cdot |\mathcal{L}|^{3.5}}\right)^{1/7} \\ &= \frac{\sqrt{\rho/d}}{20} .\end{aligned}$$

Complexity parameters. The complexity parameters of the protocol are as follows:

- *Rounds.* $2M+1 = 2 \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + 1$.
- *Proof length.* $M \cdot s + \dfrac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} |\mathcal{L}_i|$

$$\begin{aligned} &= 2 \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + \frac{d}{k^{\lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor}} + \sum_{i=1}^{\lfloor \log_k d \rfloor} \frac{|\mathcal{L}|}{2^i} \\ &\leq |\mathcal{L}| + 2 \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + k \cdot d_{\mathsf{stop}} - 1 \,. \end{aligned}$$

- *Input query complexity.* $t_0 \leq \frac{\lambda}{-\log(1-\delta')}$.
- *Proof query complexity.* Observe that $\frac{\log(1.05 \cdot \sqrt{\rho})}{\log(\sqrt{k/2})} < 0$ since $\rho < 1$. Therefore the proof query complexity is:

$$\begin{aligned} \sum_{i=1}^{M} t_i &= \sum_{i=1}^{M} \left\lceil \frac{\lambda+1}{-\log(\sqrt{\rho_i} + \eta_i)} \right\rceil \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{-\log(\sqrt{\rho_i} + \sqrt{\rho_i}/20)} \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{-\log((2/k)^{i/2} \cdot 1.05 \cdot \sqrt{\rho})} \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{i \cdot \log(\sqrt{k/2}) - \log(1.05 \cdot \sqrt{\rho})} \\ &\leq M + \frac{\lambda+1}{\log(\sqrt{k/2})} \cdot \sum_{i=1}^{M} \frac{1}{i - \frac{\log(1.05 \cdot \sqrt{\rho})}{\log(\sqrt{k/2})}} \\ &< M + \frac{\lambda+1}{\log(\sqrt{k/2})} \cdot \left(\log \left(\frac{M}{-\frac{\log(1.05 \cdot \sqrt{\rho})}{\log(\sqrt{k/2})} + 1} \right) + 1 \right) \\ &= O_k \left(\log d + \lambda \cdot \log \left(\frac{\log d}{-\log \sqrt{\rho}} \right) \right) , \end{aligned}$$

  where the final inequality follows by applying Theorem 5.9.1.

Round-by-round soundness. We begin by confirming the requirements needed in order to apply Theorem 5.4.4 using $\mathsf{B}(\rho) = \sqrt{\rho}$ as defined in Theorem 5.3.1.

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1 - \sqrt{\rho_0})$: this holds by the definition of $\delta_0 := \min\{\delta, 1 - \sqrt{\rho_0} - \eta_0\}$, since $\delta := \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])$ and $\eta_0 > 0$.
- $\delta_i \in \left(0, \min\{1 - \rho_i - 1/|\mathcal{L}_i|, 1 - \sqrt{\rho_i}\}\right)$: since $\delta_i := 1 - \sqrt{\rho_i} - \eta_i$ with $\eta_i > 0$, it holds that $\delta_i < 1 - \sqrt{\rho_i}$. Since $d \geq 4$, it holds that $1 - \rho_i - 1/|\mathcal{L}_i| = 1 - (1 + 1/d) \cdot$

$\rho_i < 1 - 1.25 \cdot \rho_i$. Finally, $1.25 \cdot \rho_i < \sqrt{\rho_i}$ holds since $\rho_i \leq 0.5$, and so $\delta_i < 1 - \sqrt{\rho_i} < 1 - \rho_i - 1/|\mathcal{L}_i|$.

- $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]$ is (δ_i, ℓ_i)-list decodable: by the Johnson bound (Theorem 2.2.5), since $\delta_i := 1 - \sqrt{\rho_i} - \eta_i$ this holds for $\ell_i = \frac{1}{2 \cdot \eta_i \cdot \sqrt{\rho_i}}$.

Now we can derive the round-by-round soundness bounds, using

$$\mathsf{err}^\star(d, \rho, \delta, \ell) := \frac{(\ell - 1) \cdot d^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{1 - \sqrt{\rho} - \delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7},$$

as in Theorem 5.3.1:

- $\varepsilon^{\mathsf{fold}}$:

$$\begin{aligned}
\varepsilon^{\mathsf{fold}} &\leq \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0) \\
&= \frac{(k-1) \cdot (d/k)^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{1 - \sqrt{\rho} - \delta', \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1) \cdot (d/k)^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{\max\left\{1 - \sqrt{\rho} - \delta, \eta_0\right\}, \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1) \cdot (d/k)^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{\eta_0, \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1) \cdot (d/k)^2}{|\mathbb{F}| \cdot (2 \cdot \eta_0)^7} \\
&\leq 2^{-\lambda},
\end{aligned}$$

  where the final inequality holds since $\eta_0 = \left(\frac{2^\lambda \cdot (k-1) \cdot (d/k)^2}{2^7 \cdot |\mathbb{F}|}\right)^{1/7}$.

- $\varepsilon_i^{\mathsf{out}}$:

$$\varepsilon_i^{\mathsf{out}} \leq \frac{d_i^s \cdot \ell_i^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s} \leq \frac{1}{4 \cdot \eta_i^2 \cdot \rho_i} \cdot \frac{d_i}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)} = \frac{1}{\eta_i^2} \cdot \frac{d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)} \leq 2^{-\lambda},$$

  where the final inequality holds since $\eta_i \geq \left(\frac{2^\lambda \cdot d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)}\right)^{1/2}$.

- $\varepsilon_i^{\mathsf{shift}}$: we first observe that $(1 - \delta_{i-1})^{t_{i-1}} = (\sqrt{\rho_i} + \eta_i)^{\left\lceil \frac{\lambda+1}{-\log(\sqrt{\rho_i} + \eta_i)} \right\rceil} \leq 2^{-\lambda-1}$. Next,

observe that:

$$\begin{aligned}
&\mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \\
&\qquad = \frac{t_{i-1} \cdot d_i^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{\eta_i, \frac{\sqrt{\rho_i}}{20}\right\}\right)^7} + \frac{(k-1) \cdot (d_i/k)^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{\eta_i, \frac{\sqrt{\rho_i}}{20}\right\}\right)^7} \\
&\qquad = \frac{t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2}{|\mathbb{F}| \cdot (2 \cdot \eta_i)^7} \\
&\qquad = 2^{-\lambda-1} .
\end{aligned}$$

The final inequality holds since $\eta_i \geq \left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2\right)}{2^7 \cdot |\mathbb{F}|}\right)^{1/7}$. Finally,

$$\begin{aligned}
\varepsilon_i^{\mathsf{shift}} &\leq (1-\delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \\
&\leq 2^{-\lambda-1} + 2^{-\lambda-1} \\
&= 2^{-\lambda} .
\end{aligned}$$

- $\varepsilon^{\mathsf{fin}}$: it holds that $\varepsilon^{\mathsf{fin}} \leq (1-\delta_M)^{t_M} = (\sqrt{\rho_M} + \eta_M)^{\left\lceil \frac{\lambda}{-\log(\sqrt{\rho_M} + \eta_M)} \right\rceil} \leq 2^{-\lambda}$.

5.9.2 Conjectured security

We bound the properties of the IOPP described in Section 5.4.3 when assuming Theorem 5.4.6. We begin by showing that $\eta_i < \rho_i/2$, then bound the complexity parameters, and finally bound round-by-round soundness error of the protocol. Note that these parameters are based on the improved ones derived in Theorem 5.4.3.

Bounds on η values. We show that since

$$|\mathbb{F}| > (\lambda+1) \cdot 2^{\lambda+2} \cdot d \cdot |\mathcal{L}|^3 \cdot \left(s + \max\left\{\left\lceil \frac{1}{-\log(1-\delta')} \right\rceil, \left\lceil \frac{1}{-\log(1.5 \cdot \rho)} \right\rceil\right\}\right) ,$$

it holds that $\eta_i \leq \rho_i/2$ for every i. First observe that

$$|\mathbb{F}| > 2^{\lambda+2} \cdot d \cdot |\mathcal{L}|^3 \cdot (s + \max\{t_0, t_i\}) ,$$

We now bound the η parameters.

- $i = 0$:

$$\eta_0 = \left(\frac{2^\lambda \cdot (k-1)^{c_2} \cdot (d/k)^{c_2}}{\rho^{c_1+c_2} \cdot |\mathbb{F}|}\right)^{1/c_1} = \frac{2^\lambda \cdot (k-1) \cdot (d/k)}{\rho^2 \cdot |\mathbb{F}|} < \frac{2^\lambda \cdot d}{\rho^2} \cdot \frac{1}{2^{\lambda+1} \cdot |\mathcal{L}|^3} < \frac{\rho}{2} .$$

- $i > 0$:

$$\eta_i := \max\left\{\begin{array}{c} \frac{2\cdot\rho_i}{d_i}, \frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s}\right)^{\frac{1}{2\cdot c_3}}, \\ \left(\frac{2^{\lambda+1}\cdot d_i^{c_2}}{\rho_i^{c_1+c_2}\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+s-1)^{c_2}+\left(\frac{k-1}{k}\right)^{c_2}\right)\right)^{1/c_1} \end{array}\right\}$$
$$= \max\left\{\frac{2\cdot\rho_i}{d_i}, \frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^2}\right)^{\frac{1}{2}}, \frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+1)+\left(\frac{k-1}{k}\right)\right)\right\}.$$

  We show that each option is bounded by $\frac{\rho_i}{2}$.

  - First option: $\frac{2\cdot\rho_i}{d_i} < \frac{\rho_i}{2}$ since $4 \le d_{\text{stop}} \le d_i$.
  - Second option:

$$\begin{aligned}\frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^2}\right)^{\frac{1}{2}} < |\mathcal{L}|\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}|)^2}\right)^{\frac{1}{2}} &\le |\mathcal{L}|\cdot\left(\frac{2^\lambda\cdot d^2}{4\cdot 2^\lambda\cdot d^2\cdot|\mathcal{L}|^6}\right)^{\frac{1}{2}} \\ &< \frac{\rho}{2\cdot d} \\ &< \frac{\rho_i}{2}.\end{aligned}$$

  - Third option:

$$\begin{aligned}\frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+1)+\left(\frac{k-1}{k}\right)\right) &< \frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot(t_{i-1}+2) \\ &< \frac{2^{\lambda+1}\cdot d\cdot|\mathcal{L}|^2\cdot(t_{i-1}+2)}{2^{\lambda+2}\cdot d\cdot|\mathcal{L}|^3\cdot(t_{i-1}+2)} \\ &= \frac{\rho}{2\cdot d} \\ &< \frac{\rho_i}{2}.\end{aligned}$$

Complexity parameters. The complexity parameters of the protocol are as follows:

- *Rounds.* $2M+1 = 2\cdot\lfloor\log_k(d/d_{\text{stop}})\rfloor + 1$.
- *Proof length.* $M\cdot s + \frac{d}{\prod_{i=0}^M k_i} + \sum_{i=1}^M |\mathcal{L}_i|$

$$\begin{aligned}&= 2\cdot s\cdot\lfloor\log_k(d/d_{\text{stop}})\rfloor + \frac{d}{k^{\lfloor\log_k(d/d_{\text{stop}})\rfloor}} + \sum_{i=1}^{\lfloor\log_k d\rfloor}\frac{|\mathcal{L}|}{2^i} \\ &\le |\mathcal{L}| + 2\cdot s\cdot\lfloor\log_k(d/d_{\text{stop}})\rfloor + k\cdot d_{\text{stop}} - 1.\end{aligned}$$

- *Input query complexity.* $t_0 \leq \frac{\lambda}{-\log(1-\delta')}$.
- *Proof query complexity.* Observe that $\frac{\log(1.5\cdot\rho)}{\log(k/2)} < 0$ since $\rho \leq 1/2$. Therefore the proof query complexity is:

$$\begin{aligned}
\sum_{i=1}^{M} t_i &= \sum_{i=1}^{M} \left\lceil \frac{\lambda+1}{-\log(\rho_i+\eta_i)} \right\rceil \\
&\leq M + (\lambda+1)\cdot \sum_{i=1}^{M} \frac{1}{-\log(\rho_i+\rho_i/2)} \\
&\leq M + (\lambda+1)\cdot \sum_{i=1}^{M} \frac{1}{-\log((2/k)^i\cdot 1.5\cdot\rho)} \\
&\leq M + (\lambda+1)\cdot \sum_{i=1}^{M} \frac{1}{i\cdot\log(k/2) - \log(1.5\cdot\rho)} \\
&\leq M + \frac{\lambda+1}{\log(k/2)}\cdot \sum_{i=1}^{M} \frac{1}{i - \frac{\log(1.5\cdot\rho)}{\log(k/2)}} \\
&< M + \frac{\lambda+1}{\log(k/2)}\cdot \left(\log\left(\frac{M}{-\frac{\log(1.5\cdot\rho)}{\log(k/2)}} + 1 \right) + 1 \right) \\
&= O_k\left(\log d + \lambda\cdot\log\left(\frac{\log d}{-\log\rho} \right) \right),
\end{aligned}$$

  where the final inequality follows by applying Theorem 5.9.1.

Round-by-round soundness. We begin by confirming the requirements needed in order to apply Theorem 5.4.4 using $\mathsf{B}(\rho) = \rho$ as defined in Theorem 5.4.6.

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1-\rho_0)$: this holds by the definition of $\delta_0 := \min\{\delta, 1-\rho_0-\eta_0\}$, since $\delta := \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])$ and $\eta_0 > 0$.
- $\delta_i \in (0,\ \min\{1-\rho_i - 1/|\mathcal{L}_i|,\ 1-\rho_i\})$: this holds since $\delta_i := 1-\rho_i-\eta_i$ with $\eta_i \geq 2/|\mathcal{L}_i|$.
- $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]$ is (δ_i, ℓ_i)-list decodable: by the Theorem 5.4.6, since $\delta_i := 1-\rho_i-\eta_i$ this holds for $\ell_i = \left(\frac{d_i}{\rho_i\cdot\eta_i}\right)^{c_3}$.

Now we can derive the round-by-round soundness bounds, using

$$\mathsf{err}^\star(d, \rho, \delta, \ell) := \frac{(\ell-1)^{c_2}\cdot d^{c_2}}{\eta^{c_1}\cdot\rho^{c_1+c_2}\cdot|\mathbb{F}|},$$

as in Theorem 5.4.6:

- $\varepsilon^{\mathsf{fold}}$:

$$\begin{aligned}\varepsilon^{\mathsf{fold}} &\leq \mathsf{err}^{\star}(d_0/k_0, \rho_0, \delta_0, k_0)\\ &= \frac{(k-1)^{c_2} \cdot (d/k)^{c_2}}{\eta_0^{c_1} \cdot \rho^{c_1+c_2} \cdot |\mathbb{F}|}\\ &\leq 2^{-\lambda}\,,\end{aligned}$$

  where the final inequality holds since $\eta_0 = \left(\frac{2^{\lambda} \cdot (k-1)^{c_2} \cdot (d/k)^{c_2}}{\rho^{c_1+c_2} \cdot |\mathbb{F}|}\right)^{1/c_1}$.

- $\varepsilon_i^{\mathsf{out}}$:

$$\varepsilon_i^{\mathsf{out}} \leq \frac{d_i^s \cdot \ell_i^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s} \leq \left(\frac{d_i}{\rho_i \cdot \eta_i}\right)^{2 \cdot c_3} \cdot \frac{d_i^s}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s} \leq 2^{-\lambda}\,,$$

  where the final inequality holds since $\eta_i \geq \frac{d_i}{\rho_i} \cdot \left(\frac{2^{\lambda} \cdot d_i^s}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s}\right)^{\frac{1}{2 \cdot c_3}}$.

- $\varepsilon_i^{\mathsf{shift}}$: we first observe that $(1-\delta_{i-1})^{t_{i-1}} = (\rho_i + \eta_i)^{\left\lceil \frac{\lambda+1}{-\log(\rho_i+\eta_i)} \right\rceil} \leq 2^{-\lambda-1}$. Next, observe that:

$$\begin{aligned}&\mathsf{err}^{\star}(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^{\star}(d_i/k_i, \rho_i, \delta_i, k_i)\\ &\qquad= \frac{(t_{i-1}+s-1)^{c_2} \cdot d_i^{c_2}}{\eta_i^{c_1} \cdot \rho_i^{c_1+c_2} \cdot |\mathbb{F}|} + \frac{(k-1)^{c_2} \cdot (d_i/k)^{c_2}}{\eta_i^{c_1} \cdot \rho_i^{c_1+c_2} \cdot |\mathbb{F}|}\\ &\qquad= \frac{d_i^{c_2}}{\eta_i^{c_1} \cdot \rho_i^{c_1+c_2} \cdot |\mathbb{F}|} \cdot \left((t_{i-1}+s-1)^{c_2} + \left(\frac{k-1}{k}\right)^{c_2}\right)\\ &\qquad= 2^{-\lambda-1}\,.\end{aligned}$$

  The final inequality holds since $\eta_i \geq \left(\frac{2^{\lambda+1} \cdot d_i^{c_2}}{\rho_i^{c_1+c_2} \cdot |\mathbb{F}|} \cdot \left((t_{i-1}+s-1)^{c_2} + \left(\frac{k-1}{k}\right)^{c_2}\right)\right)^{1/c_1}$. Finally,

$$\begin{aligned}\varepsilon_i^{\mathsf{shift}} &\leq (1-\delta_{i-1})^{t_{i-1}} + \mathsf{err}^{\star}(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^{\star}(d_i/k_i, \rho_i, \delta_i, k_i)\\ &\leq 2^{-\lambda-1} + 2^{-\lambda-1}\\ &= 2^{-\lambda}\,.\end{aligned}$$

- $\varepsilon^{\mathsf{fin}}$: it holds that $\varepsilon^{\mathsf{fin}} \leq (1-\delta_M)^{t_M} = (\rho_M + \eta_M)^{\left\lceil \frac{\lambda}{-\log(\rho_M+\eta_M)} \right\rceil} \leq 2^{-\lambda}$.

Chapter 6

A toolbox for barriers on interactive oracle proofs

6.1 Introduction

Since IOPs were invented to bypass open problems of PCPs, it is crucial to understand the limitations of IOPs, and the relation to the limitations of PCPs.

What are the limitations of IOPs, and how do they compare to PCPs?

For example: What trade-offs are there between round complexity, query complexity, and soundness error in IOPs? How small can the soundness error of an IOP be if we require constant query complexity but allow increasing the alphabet size (as in a sliding-scale PCP)?

In this chapter, we explore these and other questions. We show several results for IOPs in different regimes: (1) low-error IOPs imply low-error PCPs; (2) limitations of short IOPs; (3) limitations of high-round low-query IOPs; and (4) limitations of binary-alphabet constant-query IOPs. All these results follow from combining various tools from a new toolbox of transformations for IOPs. We discuss this toolbox in more detail in Section 6.2. We believe that our toolbox will prove useful to establish additional barriers beyond our work.

(1) Low-error IOPs imply low-error PCPs. The "sliding scale" conjecture [BGLR93] states that for every β with $1/\text{poly}(n) \leq \beta < 1$ there is a PCP system for NP that has perfect completeness, soundness error β, polynomial proof length over a $\text{poly}(1/\beta)$-size alphabet, constant query complexity, and logarithmic randomness complexity. A major open problem is constructing such PCPs when β is an inverse polynomial.

We show that (under a complexity assumption or using non-uniformity), a polylog-round IOP with inverse-polynomial soundness error and constant query complexity can be transformed into a sliding-scale PCP with inverse-polynomial soundness error.

Theorem 11 (informal)**.** *Let $\mathcal{R}$ be a relation with a public-coin IOP with perfect completeness, soundness error $1/n$, round complexity $\mathrm{polylog}(n)$, alphabet size $\mathrm{poly}(n)$, proof length $\mathrm{poly}(n)$, and query complexity $O(1)$. Then under a derandomization assumption[1] (or alternatively by using a non-uniform verifier) $\mathcal{R}$ has a PCP with perfect completeness, soundness error $1/n$, alphabet size $\mathrm{poly}(n)$, proof length $\mathrm{poly}(n)$, and query complexity $O(1)$.*

Our full theorem in Section 6.6 allows for trade-offs between the parameters of the IOP and PCP.

Theorem 11 can be interpreted as a positive result or a negative result. The positive viewpoint is that efforts towards constructing sliding-scale PCPs can rely on interaction as an additional tool. The negative viewpoint is that constructing $\mathrm{polylog}(n)$-round IOPs with sliding-scale parameters is as hard as constructing sliding-scale PCPs.

Our theorem does leave open the question of constructing $\mathrm{poly}(n)$-round IOPs with constant query complexity and small soundness error.

(2) Limitations of short IOPs. While the shortest PCPs known have quasi-linear proof length, constructing linear-size PCPs remains a major open problem. In contrast, interaction has enabled IOPs to achieve linear proof length (e.g., [BCGRS17]). Yet, we do not have a good understanding of the relation between proof length and soundness error for IOPs. We show that, under the randomized exponential-time hypothesis (RETH),[2] short IOPs for 3SAT have high soundness error.

Theorem 12 (informal)**.** *Assume* RETH *and suppose that there exists a public-coin IOP for n-variate* 3SAT *with the following parameters: perfect completeness, soundness error β, round complexity $\mathrm{polylog}(n)$, alphabet size λ, (total) proof length l, and query complexity q.*

If $\left(\frac{\mathsf{l}\cdot\log\lambda}{n}\right)^{\mathsf{q}} \leq n^{\mathrm{polylog}(n)}$, then $\beta > \Omega\left(\frac{n}{\mathsf{l}\cdot\log\lambda}\right)^{\mathsf{q}}$.

The theorem provides a barrier to improving some state-of-the-art PCPs. Dinur, Harsha, and Kindler [DHK15] come close to a sliding-scale PCP in the inverse-polynomial regime: they construct a PCP for NP with perfect completeness, soundness error $1/\mathrm{poly}(n)$, alphabet size $n^{1/\mathrm{polyloglog}(n)}$, proof length $\mathrm{poly}(n)$, and query complexity $\mathrm{polyloglog}(n)$. While IOPs have been useful in improving proof length over PCPs, Theorem 12 implies that IOPs are unlikely to help achieving nearly-linear proof length in the parameter regime of [DHK15] (even when significantly increasing alphabet size).

Corollary 6.1.1. *Assuming* RETH*, there is no public-coin IOP for n-variate* 3SAT *with perfect completeness, soundness error $1/n$, round complexity $\mathrm{polylog}(n)$, alphabet size $n^{\mathrm{polylog}(n)}$, proof length $n \cdot \mathrm{polylog}(n)$, and query complexity $\mathrm{polyloglog}(n)$.*

We leave open the question of whether IOPs in this parameter regime can be made to have linear proof length by using $O(n)$ rounds of interaction.

[1] There exists a function in E with circuit complexity $2^{\Omega(n)}$ for circuits with PSPACE gates.

[2] RETH states that there exists a constant $c > 0$ such that 3SAT $\notin$ BPTIME$[2^{c\cdot n}]$.

(3) Limitations of high-round low-query IOPs. Goldreich, Vadhan, and Wigderson [GVW02] show that $\mathrm{IP}[\mathsf{k}] \neq \mathrm{IP}[o(\mathsf{k})]$ for every k, under reasonable complexity assumptions. In other words, IPs with k rounds cannot be "compressed" to have $o(\mathsf{k})$ rounds. In contrast, Arnon, Chiesa, and Yogev [ACY22b] show that k-round IPs can be modified so that the verifier *reads* $o(\mathsf{k})$ rounds. We show that reading $o(\mathsf{k})$ rounds comes at the price of a large soundness error.

Theorem 13. *Let $L \in \mathrm{AM}[\mathsf{k}] \setminus \mathrm{AM}[\mathsf{k}']$ be a language for $\mathsf{k}' < \mathsf{k}$ and suppose that L has a public-coin IOP with perfect completeness, soundness error β, round complexity k, alphabet size $2^{\mathrm{poly}(n)}$, proof length $\mathrm{poly}(n)$, and query complexity $\mathsf{q} \leq \mathsf{k}'$. Then $\beta \geq \Omega\left(\frac{\mathsf{k}'}{\mathsf{k}}\right)^{\mathsf{q}} - n^{-c}$ for every constant $c > 0$.*

This provides a barrier to improving the parameters of IOPs in [ACY22b]. They show that any language in $\mathrm{IP}[\log(n)]$ has an IOP with perfect completeness, soundness error $1/\operatorname{polylog}(n)$, round complexity $\operatorname{polylog}(n)$, alphabet size $2^{\mathrm{poly}(n)}$, and query complexity $O(1)$. By Theorem 13 the soundness error $1/\operatorname{polylog}(n)$ is tight unless $\mathrm{IP}[\log(n)] = \mathrm{IP}[O(1)]$. Moreover, since the soundness error of IOPs is closely related to the approximation factor for the value of stochastic constraint satisfaction problems (SCSP) (see [ACY22b]), our theorem additionally provides barriers to proving hardness of approximation results for SCSPs using IOPs.

(4) Limitations of binary-alphabet constant-query IOPs. PCPs with a binary alphabet and small query complexity cannot have good soundness. In more detail, assuming the randomized exponential-time hypothesis, any binary-alphabet PCP with perfect completeness, soundness error β, and query complexity q satisfies the following.

- If $\mathsf{q} = 2$ then $\beta = 1$ (i.e., no such PCPs exist). This follows from the fact that we have linear time algorithms to check satisfiability of every binary-alphabet 2-ary constraint satisfaction problem.
- If $\mathsf{q} = 3$ then $\beta > 5/8$. Zwick [Zwi98] gives a polynomial-time algorithm that, on input a satisfiable CSP with binary alphabet and arity 3, distinguishes whether the CSP is satisfiable or whether every assignment satisfies at most a $5/8$ fraction of the constraints. This implies that, unless P = NP, every PCP for NP with binary alphabet, polynomial size, and query complexity 3 must have soundness error greater than $5/8$.[3] Håstad [Hås14] shows that this lower bound on soundness error is essentially optimal: for every $\varepsilon > 0$, he constructs a PCP for NP with perfect completeness, soundness error $5/8 + \varepsilon$, binary alphabet, polynomial proof length, and query complexity 3.

We ask whether interaction can help in further reducing the soundness error in the constant-query regime. Our next result shows that this is unlikely if the number of rounds is not large.

[3] Assuming ETH, the proof length of the PCP can be $2^{o(n)}$.

Theorem 14. *Assume* RETH *and suppose that there exists a non-adaptive public-coin IOP for* n*-variate* 3SAT *with the following parameters: perfect completeness, soundness error* β*, round complexity* k*, alphabet size* 2*, proof length* $2^{o(n)}$*, query complexity* q*, verifier randomness* r*, and verifier running time* $2^{o(n)}$*.*

- *If* $\mathsf{q} = 2$ *then* $\beta > 1 - \varepsilon$ *for every* ε *satisfying* $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$*.*
- *If* $\mathsf{q} = 3$ *then* $\beta > 5/8 - \varepsilon$ *for every* ε *satisfying* $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$*.*

For example, assuming RETH, there is no public-coin IOP with perfect completeness, soundness error $\beta = 1 - 2^{-o(n)}$, round complexity $\mathsf{k} = \text{polylog}(n)$, alphabet size 2, proof length $2^{o(n)}$, query complexity 2, and verifier randomness $\mathsf{r} = 2^{o(n)}$.

The bound on the query complexity of PCPs can be extended to q queries for any $\mathsf{q} = O(1)$ for which there is a polynomial-time algorithm that decides q-ary CSPs. Theorem 14 generalizes similarly to match the soundness error for PCPs. However, for $\mathsf{q} > 3$, we do not know the exact optimal soundness error for PCPs with perfect completeness [Has05].

Constructing an IOP for 3SAT with polynomial round complexity, binary alphabet, constant query complexity, and small soundness error remains an open problem.

6.1.1 Related work

Barriers on probabilistic proofs. We describe known limitations about PCPs, IPs, and IOPs.

- *PCPs.* If P $\neq$ NP then, for every $\mathsf{q} = o(\log n)$ and $\mathsf{r} = o(\log n)$, NP has no non-adaptive PCP with alphabet size $\lambda = O(1)$, query complexity q, and randomness complexity r. Indeed, the PCP-to-CLIQUE reduction in [FGLSS91], given an instance $\mathbb{x}$ for the language L of the PCP, produces, in polynomial time, a graph of size $\lambda^{\mathsf{q}} \cdot 2^{\mathsf{r}} \ll n$ whose maximum clique size is either large (if $\mathbb{x} \in L$) or small (if $\mathbb{x} \notin L$), where the gap between these sizes depends on the PCP's completeness and soundness errors. By iteratively applying that reduction a polynomial number of times, one can (in polynomial time) reduce $\mathbb{x}$ to a graph G of size $O(\log n)$, while preserving the large-or-small property of the maximum clique. Since the size of G is logarithmic, one can then determine in polynomial time whether the largest clique in G is large or small, and thereby decide membership for the original instance $\mathbb{x}$.

  Moreover, if P $\neq$ NP then NP does not have non-adaptive PCPs with alphabet size $\lambda = O(1)$, query complexity $\mathsf{q} = O(1)$, and randomness complexity $\mathsf{r} = O(\log n)$ with soundness error $\beta < \frac{\log \lambda}{\lambda^{\mathsf{q}-1}}$. Indeed, such a non-adaptive PCP can be converted into a CSP of size $\text{poly}(n)$, and any efficient algorithm for approximating the CSP's number of satisfied constraints imposes a limitation on the soundness error β. For example, the bound $\frac{\log \lambda}{\lambda^{\mathsf{q}-1}}$ follows from the approximation algorithm in [MNT22]. Assuming ETH, these limitations can be extended to PCPs with super-polynomial

proof length and super-constant alphabet size and query complexity. For completeness, in Section 6.9.1 we show quantitatively how to combine PCPs with small soundness error for 3SAT and polynomial-time approximation algorithms for CSPs in order to decide 3SAT faster than is possible under ETH.

Notice that an adaptive PCP with alphabet size λ and query complexity q can be converted into a non-adaptive PCP with query complexity λ^q, which is constant when $\lambda = O(1)$ and $q = O(1)$. Hence the above discussion applies to adaptive PCPs in this regime as well.

- *IPs.* [GH98] show that public-coin IPs with bounded prover communication complexity can be decided in non-trivial (probabilistic) time. [GVW02] strengthen these results for the case of private-coin IPs, showing that similar bounds on communication imply that the complement of the language can be decided in non-trivial non-deterministic time. Such results are limitations on IPs for languages believed to be hard, such as SAT.

- *IOPs.* In order to derive barriers for succinct arguments, [CY20] extend to IOPs the limitations of [GH98], showing barriers for IOPs with small soundness error relative to query complexity.

  [NR22] show limitations for *succinct* IOPs for circuit SAT (CSAT), where the proof length is polynomial in the number n of circuit inputs. The results cover different parameters, depending on the "plausibility" of the complexity assumption used. For example (on the most probable end), suppose that the satisfiability of a circuit C cannot be decided by a $\text{poly}(n)$-space algorithm following $\text{poly}(|C|)$-time preprocessing. Then there is no succinct IOP for CSAT with constant round complexity and logarithmic query complexity.

IOP-to-IOP transformations. Our toolbox (outlined in Section 6.2) contains IOP-to-IOP transformations that include round reduction, achieving perfect completeness, and derandomization.

- [ACY22a; ACY22b] provide IOP-to-IOP transformations for round reduction and achieving perfect completeness, but we cannot use them because those transformations do *not* preserve query complexity of the IOP (a key property for us).

- [NR22] show that any public-coin IOP can be transformed into one with less interaction randomness at the cost of introducing a "common reference string" (CRS) and satisfying only non-adaptive soundness. Their main goal is to achieve randomness complexity that depends (logarithmically) only on the prover-to-verifier communication complexity (but not the instance length) and on an error parameter over the choice of the CRS. They also show that the CRS can be replaced with non-uniform advice for the verifier at the cost of increasing the randomness complexity to also depend (logarithmically) on the instance length. Our derandomization lemma focuses on IOPs with a non-uniform verifier and allows choosing the

target randomness complexity, rather than optimizing with regards to the prover-to-verifier communication complexity.

- [AG21] show how to derandomize *private-coin IPs* via non-uniform advice or PRGs. Our derandomization lemma applies to public-coin IOPs.

6.2 Techniques

We describe our tools for IOPs and sketch their proofs, and then show how they can be applied to achieve our main results. The tools are divided into three groups.

1. **Tools for length and round reduction:** Section 6.2.1 outlines transformations that decrease the length and round complexity of IOPs with low query complexity. Details are in Section 6.3.
2. **Tools for improving completeness:** Section 6.2.2 outlines transformations that improve the completeness errors of IOPs. Details are in Section 6.4.
3. **Tools for derandomization:** Section 6.2.3 outlines transformations that decrease the number of random bits used by the IOP verifier. Details are in Section 6.5.

Following the presentation of our toolbox, in Section 6.2.4 we explain how we use the tools (in conjunction with additional arguments) to derive the theorems described in Section 6.1.

6.2.1 Tools for length and round reduction

We describe how to decrease the length and round complexity of IOPs.

Lemma 5 (informal)**.** *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ with completeness error α, soundness error β, round complexity k, alphabet size λ, per-round proof length l, query complexity q, per-round verifier randomness r, and verifier running time vt.*

1. ***Length reduction:*** *Let ℓ be a parameter with $\mathsf{q} \leq \ell \leq \mathsf{k} \cdot \mathsf{l}$. Then $\mathcal{R}$ has a public-coin IOP with completeness error $1 - (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k} \cdot \mathsf{l}))^{\mathsf{q}}$, soundness error β, round complexity k, alphabet size λ, **total proof length $\boldsymbol{\ell}$**, query complexity q, per-round verifier randomness $\mathsf{r} + \ell \cdot \log(\mathsf{k} \cdot \mathsf{l})$, and verifier running time $\mathrm{poly}(\mathsf{vt}, \ell)$.*
2. ***Round reduction:*** *Let k' be a parameter with $\mathsf{q} \leq \mathsf{k}' \leq \mathsf{k}$. Then $\mathcal{R}$ has a public-coin IOP with completeness error $1 - (1-\alpha) \cdot (\mathsf{k}'/(e \cdot \mathsf{k}))^{\mathsf{q}}$, soundness error β, **round complexity $\boldsymbol{\mathsf{k}' + 1}$**, alphabet size λ, per-round proof length l, query complexity q, per-round verifier randomness $\mathsf{k} \cdot (\mathsf{r} + \log \mathsf{k})$, and verifier running time $\mathrm{poly}(\mathsf{vt})$.*
3. ***Unrolling to PCP:*** *$\mathcal{R}$ has a PCP with completeness error α, soundness error β, alphabet size λ, proof length $\mathsf{l} \cdot 2^{O(\mathsf{k} \cdot \mathsf{r})}$, query complexity q, randomness $\mathsf{k} \cdot \mathsf{r}$, and verifier running time $\mathrm{poly}(\mathsf{vt})$.*

Below we sketch the proofs of Items 1 and 2. Item 3 is folklore and follows by setting the PCP to equal the interaction tree of the IOP.

Length reduction. The length of low-query IOPs can be reduced while incurring an increase in the completeness error. The intuition is that if the IOP has query complexity $\mathsf{q} \ll \mathsf{k} \cdot \mathsf{l}$, then each symbol in the proof is read by the verifier with small probability. Hence, if the prover omits a random subset of the proof symbols, the verifier is unlikely to require these missing symbols.

Construction 6.2.1 (informal)**.** The new prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{V}'$ *guesses the locations that* $\mathbf{V}$ *will query.* $\mathbf{V}'$ samples and sends a random set $I \subseteq [\mathsf{k} \cdot \mathsf{l}]$ of ℓ indices from among all the prover message symbols.
2. *The original IOP is simulated with prover messages omitted according to* I*.* For every $j \in [\mathsf{k}]$:
   (a) $\mathbf{V}'$ sends $\alpha_j \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) $\mathbf{P}'$ computes $\Pi_j := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$ and sends Π'_j equal to Π_j with symbols outside of I omitted.
3. $\mathbf{V}'$ *simulates* $\mathbf{V}$*, and rejects if any queries are made outside of* I*.* $\mathbf{V}'$ simulates the decision stage of $\mathbf{V}$ given input $\mathbb{x}$. Whenever an index $i \in I$ is queried, return the appropriate symbol from the prover messages. If an index $i \notin I$ is queried, then immediately reject. Output the same answer as $\mathbf{V}$.

The *total* proof length is ℓ since the prover $\mathbf{P}'$ sends only those symbols whose index is in I (which has size ℓ). The per-round verifier randomness at most $\mathsf{r} + \ell \cdot \log(\mathsf{k} \cdot \mathsf{l})$ because in the first round the verifier sends I (which can be described with $\ell \cdot \log(\mathsf{k} \cdot \mathsf{l})$ random bits) and then it sends its first message of r bits. The rest of the complexity parameters follow straightforwardly from the construction.

Soundness follows from the fact that the changes made to the IOP can only increase the chance that the verifier rejects. We sketch the proof of completeness. Fix some $\mathbb{x} \in L$. The locations read by $\mathbf{V}$ are independent of the set I. Therefore, the probability that $\mathbf{V}$ queries outside the set I is $\binom{\mathsf{k}\cdot\mathsf{l}-\mathsf{q}}{\ell-\mathsf{q}} / \binom{\mathsf{k}\cdot\mathsf{l}}{\ell} \geq (\ell/(e \cdot \mathsf{k} \cdot \mathsf{l}))^{\mathsf{q}}$. Conditioned on $\mathbf{V}$ querying only inside I, $\mathbf{V}$ accepts with probability at least $1 - \alpha$. Hence the probability that the new verifier $\mathbf{V}'$ accepts is at least $(1-\alpha) \cdot (\ell/(e \cdot \mathsf{k} \cdot \mathsf{l}))^{\mathsf{q}}$.

Round reduction. We sketch how the round-complexity of low-query IOPs can be reduced. The intuition behind this lemma is similar to that described for length reduction: if $\mathsf{q} \ll \mathsf{k}$, then the verifier is unlikely to need most of the rounds, so removing a random subset of the rounds does not harm completeness by much. Below we describe the transformation for IOP round reduction.

Construction 6.2.2 (informal)**.** The new prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{V}'$ *guesses the rounds that* $\mathbf{V}$ *will query.* $\mathbf{V}'$ samples and sends a random set $I \subseteq [\mathsf{k}]$ of k' indices. Denote $I := (i_1, \ldots, i_{\mathsf{k}'})$ with $i_j < i_{j+1}$ and let $i_0 := 1$.
2. *The original IOP is simulated with rounds omitted according to* I*.* For every $j \in [\mathsf{k}']$:
   (a) $\mathbf{V}'$ sends $\alpha_{i_{(j-1)}+1}, \ldots, \alpha_{i_j} \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) $\mathbf{P}'$ computes and sends $\Pi_j := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i_j})$.
3. $\mathbf{V}'$ *simulates* $\mathbf{V}$*, and rejects if any queries are made outside of* I*.* $\mathbf{V}'$ samples $\alpha_{i_{\mathsf{k}'}+1}, \ldots, \alpha_{\mathsf{k}} \leftarrow \{0,1\}^{\mathsf{r}}$ simulates the decision stage of $\mathbf{V}$ given input $\mathbb{x}$ and verifier messages $\alpha_1, \ldots, \alpha_{\mathsf{k}}$. Whenever an index in round $i \in I$ is queried, return

the appropriate symbol in the prover messages. If a round $i \notin I$ is queried, then immediately reject. Output the same answer as $\mathbf{V}$.

A technical remark: as written above, the protocol is not public-coin because the verifier's first message I dictates the length of subsequent verifier messages. Nevertheless, the protocol can be made public-coin by padding verifier messages to $\mathsf{k} \cdot \mathsf{r}$ bits. The prover and verifier act as in the protocol description, ignoring the padding bits. The verifier additionally sends $\mathsf{k}' \cdot \log \mathsf{k}$ bits as the choice of the set I. Thus, the per-round randomness of the verifier is $\mathsf{k} \cdot \mathsf{r} + \mathsf{k}' \cdot \log \mathsf{k} \leq \mathsf{k} \cdot (\mathsf{r} + \log \mathsf{k})$.

6.2.2 Tools for improving completeness

A transformation for achieving perfect completeness for IPs is shown in [FGMSZ89]. Directly applying that transformation to IOPs increases the query complexity of the protocol significantly. We show a variant of the transformation in [FGMSZ89] that preserves query complexity up to a small additive constant.

Lemma 6 (informal)**.** *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ with completeness error α, soundness error β, round complexity k, alphabet size λ, per-round proof length l, query complexity q, per-round verifier randomness r, and verifier running time vt.*

*Then $\mathcal{R}$ has a public-coin IOP with **perfect completeness**, soundness error $O\left(\frac{\beta \cdot \mathsf{k} \cdot \mathsf{r}}{\log(1/\alpha)}\right)$, round complexity $\mathsf{k}+1$, alphabet size $\max\{\lambda, 2^{\mathsf{k} \cdot \mathsf{r}}\}$, per-round proof length $O\left(\frac{\mathsf{l} \cdot \mathsf{k} \cdot \mathsf{r}}{\log(1/\alpha)}\right)$, query complexity $\mathsf{q}+2$, per-round verifier randomness r, and verifier running time $\mathrm{poly}(\mathsf{vt})$.*

Remark 6.2.3. If only small completeness error is desired (rather than completeness error 0), then this can be achieved with similar query complexity but smaller overhead to the alphabet size. See Section 6.4.1 for more details.

Review: perfect completeness for IPs. Consider the set S of verifier random coins $\vec{\alpha} = (\alpha_1, \ldots, \alpha_{\mathsf{k}})$ (over the entire protocol) where the honest prover has a strategy to make the verifier accept if it is sent these strings while interacting with the verifier. Given the matching prover messages, the verifier can efficiently check whether $\vec{\alpha} \in S$. [FGMSZ89] shows that for large enough t there exist "shifts" $\vec{z}_1, \ldots, \vec{z}_t$ such that for *every* choice of verifier randomness $\vec{\alpha}$ there exists j such that $(\vec{z}_j \oplus \vec{\alpha}) \in S$. It follows that the honest prover needs only to send these shifts, and then run the protocol with the verifier, giving answers matching each shift. At the end of the protocol, the verifier accepts if and only if $\vee_{j=1}^{t}\left((\vec{z}_j \oplus \vec{\alpha}) \stackrel{?}{\in} S\right) = 1$. The soundness error degrades by a multiplicative factor of t since a malicious prover only needs to convince the verifier in one execution.

Perfect completeness for IOPs. The aforementioned verifier computes the "OR" of t expressions. We observe that, in order to prove the claim $\vee_{j=1}^{t}\left((\vec{z}_j \oplus \vec{\alpha}) \stackrel{?}{\in} S\right) = 1$, it suffices for the prover to send the verifier a *single* index j where $(\vec{z}_j \oplus \vec{\alpha}) \in S$, which

is then checked by the verifier. The verifier only needs to check *a single execution* of the IOP, rather than t, and so the query complexity of the protocol is preserved up to reading the index j and shift $\vec{z}_j$.

Construction 6.2.4. Let $t := 2 \cdot \left(\frac{\mathsf{r} \cdot \mathsf{k}}{\log(1/\alpha)}\right)$. The new prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{P}'$ *sends t "shifts" for the verifier randomness.* $\mathbf{P}'$ sends

$$\vec{z}_1, \ldots, \vec{z}_t = (z_{1,1}, \ldots, z_{1,\mathsf{k}}), \ldots, (z_{t,1}, \ldots z_{t,\mathsf{k}}) \in \{0,1\}^{\mathsf{r} \cdot \mathsf{k}},$$

   to the verifier such that for every $\vec{\alpha}$ there exists j where $(\vec{z}_j \oplus \vec{\alpha}) \in S$ (i.e., the original prover $\mathbf{P}$ has an accepting strategy for verifier randomness $(\vec{z}_j \oplus \vec{\alpha})$).
2. *Original IOP is simulated, where for every verifier message, prover replies with a message for each shifted randomness.* For $i = 1, \ldots, \mathsf{k}$:
   - $\mathbf{V}'$: Choose $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}}$ uniformly and send to the prover.
   - $\mathbf{P}'$: Send $\{\Pi_{j,i}\}_{j \in [t]}$ where $\Pi_{j,i} := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha_i \oplus z_{j,i})$.
3. *Prover sends index j of shift where its messages succeed in convincing the verifier.* $\mathbf{P}'$: If there exists an index $j \in [t]$, such that $\mathbf{V}^{\Pi_{j,1},\ldots,\Pi_{j,\mathsf{k}}}(\mathbb{x}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha_{\mathsf{k}} \oplus z_{j,\mathsf{k}}) = 1$, then send j to the verifier $\mathbf{V}'$ as a non-oracle message. Otherwise, send $\perp$.
4. $\mathbf{V}'$ *checks that* $\mathbf{V}$ *accepts the "shifted" j-th execution.* $\mathbf{V}'$: Receive j as a non-oracle message.
   (a) If $j = \perp$, then reject.
   (b) Otherwise, query $\vec{z}_j = (z_{j,1}, \ldots, z_{j,\mathsf{k}})$ and check that

$$\mathbf{V}^{\Pi_{j,1},\ldots,\Pi_{j,\mathsf{k}}}(\mathbb{x}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha_{\mathsf{k}} \oplus z_{j,\mathsf{k}}) = 1,$$

   querying the appropriate proofs as required by $\mathbf{V}$.

6.2.3 Tools for derandomization

We show how to derandomize public-coin IOPs based on non-uniform advice or based on pseudorandom generators (PRGs), while preserving the use of public-coins. Both transformations achieve logarithmic randomness complexity but slightly increase completeness and soundness error. Round complexity, proof length, and query complexity are preserved.

Lemma 7 (informal)**.** *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ with completeness error α, soundness error β, round complexity k, alphabet size λ, per-round proof length l, query complexity q, per-round verifier randomness r, and verifier running time vt.*

1. ***Derandomization using PRGs:*** *Suppose that there exists a PRG against polynomial-size* PSPACE *circuits with seed length ℓ, error ε and evaluation time $\mathsf{t}_{\mathsf{PRG}}$. Then $\mathcal{R}$ has a public-coin IOP with completeness error $1 - O((1-\alpha) - \varepsilon \cdot \mathsf{k}^2)$, soundness error $O(\beta + \varepsilon \cdot \mathsf{k}^3)$,*

round complexity k*, alphabet size* λ*, per-round proof length* l*, query complexity* q*,* ***per-round verifier randomness*** $\boldsymbol{\ell}$*, and verifier running time* $\mathrm{poly}(\mathsf{vt}, \mathsf{t}_{\mathsf{PRG}})$*.*

(Such a PRG with seed length $\ell = O(\log|\mathbb{x}|)$*, error* $\varepsilon = 1/\mathrm{poly}(|\mathbb{x}|)$ *and computation time* $\mathsf{t}_{\mathsf{PRG}} = \mathrm{poly}(|\mathbb{x}|)$ *exists if there exists a function in* E *with circuit complexity* $2^{\Omega(n)}$ *for circuits with* PSPACE *gates.)*

2. ***Derandomization using non-uniformity:*** *Let* $\varepsilon \in (0,1)$ *be a parameter. Then* $\mathcal{R}$ *has a public-coin IOP with completeness error* $\alpha + \mathsf{k}\cdot\varepsilon$*, soundness error* $\beta + \mathsf{k}\cdot\varepsilon$*, round complexity* k*, alphabet size* λ*, per-round proof length* l*, query complexity* q*,* ***per-round verifier randomness*** $\boldsymbol{\Theta(\log((\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon))}$*, and verifier running time* $\mathrm{poly}(\mathsf{vt},\mathsf{k},\mathsf{l},\mathsf{r},1/\varepsilon)$*, where the verifier receives* $\mathrm{poly}(|\mathbb{x}|,\mathsf{k},\mathsf{r},1/\varepsilon)$ *bits of non-uniform advice. Moreover, a random string constitutes good advice with probability* $1-2^{-|\mathbb{x}|}$*.*

We focus the overview below on Item 1. Item 2 can be shown in a similar manner.

Derandomization using PRGs. We show that IOPs can be derandomized using a pseudo-random generator. In this transformation, the verifier samples seeds for the PRG rather than uniform random messages. Thus the verifier randomness per-round is as small as a seed of the PRG.

Construction 6.2.5 (informal)**.** On instance $\mathbb{x}$ and witness $\mathbb{w}$, the protocol $(\mathbf{P}', \mathbf{V}')$ proceeds as follows:

1. *Simulate original IOP where verifier messages are chosen using the PRG.* For $j = 1, \ldots, \mathsf{k}$:
   (a) $\mathbf{V}'$: Sample and send a random $\alpha_j \leftarrow \{0,1\}^{\ell}$.
   (b) $\mathbf{P}'$: Compute and send the prover message Π_j that maximizes the probability that $\mathbf{V}$ accepts where all of the verifier messages are chosen using the PRG G.
2. $\mathbf{V}'$: Accept if and only if $\mathbf{V}^{\Pi_1,\ldots,\Pi_{\mathsf{k}}}(\mathbb{x}, \mathsf{G}(\alpha_1), \ldots, \mathsf{G}(\alpha_{\mathsf{k}})) = 1$.

The verifier sends ℓ_{PRG} bits of randomness in each round, since it sends a seed for the PRG. The rest of the complexity parameters follow straightforwardly from the construction.

Interaction trees. The *interaction tree* of a protocol on input $\mathbb{x}$, denoted $T_{\mathbb{x}}$ is the full tree of all possible transcripts corresponding to each choice of prover and verifier messages. The leaves are labelled as accepting or rejecting corresponding to whether the verifier accepts or rejects the full transcript represented by the leaf.

The *value* of an interaction tree $T_{\mathbb{x}}$, denoted by $\mathsf{val}(T_{\mathbb{x}})$, is the probability of reaching an accepting leaf from the root of the tree in a walk on the tree where verifier messages are chosen uniformly at random and prover messages are chosen so as to maximize the probability of reaching an accepting node. The notion of value extends to sub-trees as well, where the value is the probability of reaching an accepting leaf when beginning on the root of the sub-tree. Notice that $\mathsf{val}(T_{\mathbb{x}}) = \max_{\tilde{\mathbf{P}}}\{\Pr[\langle\tilde{\mathbf{P}}, \mathbf{V}\rangle(\mathbb{x}) = 1]\}$. Moreover, $\mathsf{val}(T_{\mathbb{x}})$ can be computed in space that is polynomial in $|\mathbb{x}|$, the round complexity, the proof length, and the verifier randomness of the IOP.

Completeness and soundness. Completeness and soundness follow straightforwardly from Theorem 6.2.6, which says that the value of the interaction tree of the IOP does not change by much when the verifier messages are sampled via a PRG.

Claim 6.2.6. *Let* G *be a PRG against circuits of size* $\mathrm{poly}(|\mathbb{x}|)$ *with* PSPACE *gates. Then for every instance* $\mathbb{x}$*:*

$$O(\mathsf{val}(T) - \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2) \leq \mathsf{val}(T_{\mathsf{G}}) \leq O(\mathsf{val}(T) + \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3)\,,$$

where T *is the interaction tree of the IOP and* T_{G} *is the interaction tree of* $(\mathbf{P}', \mathbf{V}')$*, which is identical to* T *except verifier randomness is always sampled using the PRG* G*.*

We give a simplified sketch of the proof of the claim. Let $T^{(0)} := T_{\mathsf{G}}$ and for $i = 1, \ldots, \mathsf{k}$ let $T^{(i)}$ be the tree of an intermediate protocol where the messages $\alpha_1, \ldots, \alpha_i$ are chosen uniformly at random and $\alpha_{i+1}, \ldots, \alpha_{\mathsf{k}}$ are chosen from the PRG. Notice that $T^{(\mathsf{k})} = T$.

We show that, under a simplifying assumption to be described later, there exist circuit families $\mathsf{Circ}^{(1)}, \ldots, \mathsf{Circ}^{(\mathsf{k})}$ each comprised of circuits of size $\mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ that have PSPACE gates, such that if G fools $\mathsf{Circ}^{(i)}$ then

$$|\mathsf{val}(T^{(i-1)}) - \mathsf{val}(T^{(i)})| \leq \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}\,.$$

Letting $\mathsf{Circ} := \cup_i \mathsf{Circ}_i$, we have that if G fools Circ (i.e., fools circuits of size $\max_{C \in \mathsf{Circ}} |C| = \mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$), then

$$|\mathsf{val}(T_{\mathbb{x}}) - \mathsf{val}(T_{\mathbb{x},\mathsf{G}})| \leq \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2\,.$$

Fix some i. We show a family $\mathsf{Circ}^{(i)}$ such that if G fools $\mathsf{Circ}^{(i)}$ then $|\mathsf{val}(T^{(i-1)}) - \mathsf{val}(T^{(i)})| \leq \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}$. Consider a fixed node in $T^{(i)}$ corresponding to the transcript prefix $\mathsf{tr} = (\alpha_1, m_1, \ldots, \alpha_{i-1}, m_{i-1})$ (which is empty if $i = 1$). For α_i let $T^{(i,\mathsf{tr})}(\alpha_i)$ be the sub-tree of $T^{(i)}$ whose root corresponds to the transcript $(\mathsf{tr}||\alpha_i)$.

Define

$$S := \left\{ \left(1 + \frac{1}{3\mathsf{k}}\right)^{-1}, \ldots, \left(1 + \frac{1}{3\mathsf{k}}\right)^{-O(\mathsf{k})}, 0 \right\}.$$

We make the simplifying assumption that $\mathsf{val}(T^{(i,\mathsf{tr})}(\alpha_i)) \in S$ and $\mathsf{val}(T^{(i-1,\mathsf{tr})}(\alpha_i)) \in S$ for every α_i. In the full proof of the claim we achieve this by discretizing the functions $\mathsf{val}(T^{(i,\mathsf{tr})}(\cdot))$ and $\mathsf{val}(T^{(i-1,\mathsf{tr})}(\cdot))$, which incurs additional errors. For simplicity, we ignore these errors in this overview.

For every transcript tr, let $\mathsf{Circ}^{(i,\mathsf{tr})} := \{C_p^{(i,\mathsf{tr})}\}_{p \in S}$ where each circuit $C_p^{(i,\mathsf{tr})}$, on input α_i, outputs 1 if and only if $\mathsf{val}(T^{(i,\mathsf{tr})}(\alpha_i)) = p$. We observe that a careful implementation of $C_p^{(i,\mathsf{tr})}$ (computing the value of a tree can be done space proportional to

its depth) has size at most $\mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ using PSPACE gates. Thus, if G fools every circuit in the family $\mathsf{Circ}^{(i,\mathsf{tr})}$ we get that

$$\begin{aligned}
\mathsf{val}(T^{(i-1,\mathsf{tr})}) &= \sum_{p \in S} p \cdot \Pr_{s}[\mathsf{C}_p^{(i,\mathsf{tr})}(\mathsf{G}(s)) = 1] \\
&\leq \sum_{p \in S} p \cdot \left(\Pr_{\alpha_i}[\mathsf{C}_p^{(i,\mathsf{tr})}(\alpha_i) = 1] + \epsilon_{\mathsf{PRG}} \right) \\
&= \mathsf{val}(T^{(i,\mathsf{tr})}) + \sum_{p \in S} p \cdot \epsilon_{\mathsf{PRG}} \\
&\leq \mathsf{val}(T^{(i,\mathsf{tr})}) + O(\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}) \,,
\end{aligned}$$

where $T^{(i-1,\mathsf{tr})}$ is the sub-tree of $T^{(i-1)}$ whose root corresponds to the transcript tr. The final inequality follows by the fact that $\sum_{p \in S} p = \sum_{i=1}^{O(\mathsf{k})} (1 + 1/3\mathsf{k})^{-i}$ is a geometric series bounded by $O(\mathsf{k})$.

We can similarly show that $\mathsf{val}(T^{(i-1,\mathsf{tr})}) \geq \mathsf{val}(T^{(i,\mathsf{tr})}) - O(\epsilon_{\mathsf{PRG}} \cdot \mathsf{k})$. Notice that $\mathsf{val}(T^{(i)}) = \mathbb{E}_{\mathsf{tr}}[\mathsf{val}(T^{(i,\mathsf{tr})})]$ and $\mathsf{val}(T^{(i-1)}) = \mathbb{E}_{\mathsf{tr}}[\mathsf{val}(T^{(i-1,\mathsf{tr})})]$ (where the expectation is over the verifier's random coins). Therefore, if the G fools the entire circuit family $\mathsf{Circ}^{(i)} := \cup_{\mathsf{tr}} \mathsf{Circ}^{(i,\mathsf{tr})}$ then we have

$$\begin{aligned}
|\mathsf{val}(T^{(i-1)}) - \mathsf{val}(T^{(i)})| &= |\mathbb{E}_{\mathsf{tr}}[\mathsf{val}(T^{(i-1,\mathsf{tr})})] - \mathbb{E}_{\mathsf{tr}}[\mathsf{val}(T^{(i,\mathsf{tr})})]| \\
&\leq \left| \mathbb{E}_{\mathsf{tr}} \left[\mathsf{val}(T^{(i,\mathsf{tr})}) + O(\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}) \right] - \mathbb{E}_{\mathsf{tr}} \left[\mathsf{val}(T^{(i,\mathsf{tr})}) \right] \right| \\
&= O(\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}) \,.
\end{aligned}$$

6.2.4 Deriving our results using the tools

We use the toolbox developed in the previous sections to derive the theorems in Section 6.1. Each theorem is proved by applying a carefully chosen sequence of tools (along with other arguments). Figure 6.1 summarizes which tools are used to derive each theorem and the order of their use.

6.2.4.1 Low-error IOPs to low-error PCPs

We sketch the proof of Theorem 11, which shows that low-error IOPs can be transformed into low-error PCPs. The proof is a sequence of transformations from our toolbox, whose goal is to transform the IOP into one that is efficient enough to be unrolled into a PCP via Item 3 of Lemma 5. This unrolling has an exponential dependency on the round complexity and on the verifier randomness complexity of the IOP, so we seek to decrease these without increasing the soundness error.

Decreasing the round complexity is done using the round-reduction transformation of Lemma 5, and decreasing the verifier randomness is done using either one of

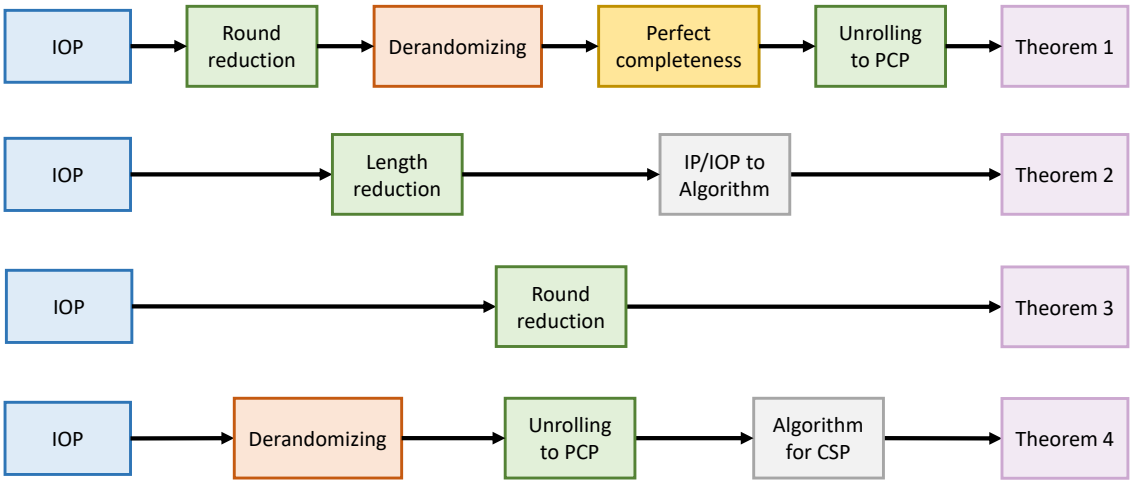


Figure 6.1: Summary of how our tools are used to derive each theorem. Grey boxes are prior work.

our derandomization lemmas (Lemma 7). Since both transformations degrade completeness, prior to applying the unrolling lemma (Item 3 of Lemma 5), we restore the IOP back to having perfect completeness using Lemma 6. Since the transformation for perfect completeness increases the soundness error, we counterbalance it by beginning the sequence of transformations with a small number of parallel repetitions.

In somewhat more detail, the sequence of transformations is as follows.

1. **Initial IOP.** We begin with an IOP with the following parameters: perfect completeness, soundness error $1/|\mathbb{x}|$, round complexity $\text{polylog}(|\mathbb{x}|)$, alphabet size $\text{poly}(|\mathbb{x}|)$, proof length $\text{poly}(|\mathbb{x}|)$, query complexity $O(1)$, and per-round randomness $\text{poly}(|\mathbb{x}|)$.

2. **Parallel repetition.** Repeat the protocol twice in parallel, and have the verifier accept if and only if both executions are accepted. This yields a public-coin IOP for $\mathcal{R}$ with: perfect completeness, **soundness error $1/|\mathbb{x}|^2$**, round complexity $\mathsf{k} = \text{polylog}(|\mathbb{x}|)$, alphabet size $\text{poly}(|\mathbb{x}|)$, query complexity $\mathsf{q} = O(1)$, and per-round randomness $\text{poly}(|\mathbb{x}|)$.

3. **Round reduction.** Reduce the number of rounds of the IOP via Item 2 of Lemma 5 with $\ell := \mathsf{q}$ where $\mathsf{q} = O(1)$ is the query complexity of the IOP verifier. This transformation results in a public-coin IOP for $\mathcal{R}$ with: completeness error $1 - (\mathsf{q}/(e \cdot \mathsf{k}))^{\mathsf{q}} = 1 - 1/\text{polylog}(|\mathbb{x}|)$, soundness error $1/|\mathbb{x}|^2$, **round complexity $O(1)$**, alphabet size $\text{poly}(|\mathbb{x}|)$, query complexity $O(1)$, and per-round randomness $\text{poly}(|\mathbb{x}|)$.

4. **Derandomization.** Derandomize the IOP verifier using either item of Lemma 7. This results in a public-coin IOP for $\mathcal{R}$ with: completeness error $1 - 1/\text{polylog}(|\mathbb{x}|)$, soundness error $O(1/|\mathbb{x}|^2)$, round complexity $O(1)$, alphabet size $\text{poly}(|\mathbb{x}|)$, query complexity $O(1)$, and **per-round randomness $O(\log |\mathbb{x}|)$**.

5. **Perfect completeness.** Improve the IOP to have perfect completeness using Lemma 6. The resulting IOP has the following parameters: **perfect completeness**, soundness error

$$O(1/|\mathbb{x}|^2) \cdot \left(\frac{\mathsf{q} \cdot O(\log |\mathbb{x}|)}{-\log(1 - 1/\operatorname{polylog}(|\mathbb{x}|))} \right) \leq 1/|\mathbb{x}|\,,$$

round complexity $O(1)$, alphabet size $\operatorname{poly}(|\mathbb{x}|)$, query complexity $O(1)$, and randomness $O(\log |\mathbb{x}|)$.

6. **Unrolling to PCP.** Unroll the IOP with perfect completeness into a PCP via Item 3 of Lemma 5. This gives us our final PCP with parameters: perfect completeness, soundness error $1/|\mathbb{x}|$, alphabet size $\operatorname{poly}(|\mathbb{x}|)$, proof length $\operatorname{poly}(|\mathbb{x}|)$, query complexity $O(1)$, and randomness complexity $O(\log |\mathbb{x}|)$.

6.2.4.2 Limitations of short IOPs

We sketch the proof of Theorem 12, which shows that short IOPs with small soundness contradict RETH, the hypothesis that 3SAT $\notin$ BPTIME$[2^{c \cdot n}]$ for a constant $c > 0$. First, we convert the IOP into a short IP, and then apply a transformation from [CY20] that converts short IPs into fast probabilistic algorithms. This leads to a fast algorithm for 3SAT, contradicting RETH.

Consider a public-coin IOP for n-variate 3SAT with parameters as in Theorem 12: perfect completeness, soundness error β, round complexity $\operatorname{polylog}(n)$, alphabet size λ, (total) proof length l, query complexity q, and verifier randomness $\operatorname{poly}(n)$. Suppose towards contradiction that $\mathsf{l} \geq n$ and $\left(\frac{\mathsf{l} \cdot \log \lambda}{n}\right)^{\mathsf{q}} \leq n^{\operatorname{polylog}(n)}$ and that $\beta = \frac{1}{2} \cdot \left(\frac{2 \cdot e \cdot \mathsf{l} \cdot \log \lambda}{c \cdot n}\right)^{-\mathsf{q}} \geq n^{-\operatorname{polylog}(n)}$.[4]

We apply the following transformations.

1. **Length reduction.** Apply Item 1 of Lemma 5 with parameter $\ell := e \cdot \mathsf{l} \cdot (2\beta)^{1/\mathsf{q}}$. This results in an IOP with: completeness error $\alpha' := 1 - 2\beta$, soundness error β, round complexity $\mathsf{k}' := \operatorname{polylog}(n)$, alphabet size $\lambda' := \lambda$, and **proof length $\mathsf{l}' := e \cdot \mathsf{l} \cdot (2\beta)^{1/\mathsf{q}}$**.

2. **IOP to algorithm.** Convert the IOP into an algorithm using a lemma from [CY20] that says that if a relation $\mathcal{R}$ has a public-coin IP with completeness error α', soundness error β', round complexity k', and prover-to-verifier communication length l' of symbols of size λ', then there is a probabilistic algorithm for deciding $\mathcal{R}$ in time $2^{O(d)+o(n)}$ for $d := \mathsf{l}' \cdot \log \lambda' + \mathsf{k}' \cdot \log \frac{\mathsf{k}'}{1-\alpha'-\beta'}$. Notice that while the result from [CY20] applies to IPs rather than IOPs, one can straightforwardly convert an IOP into an IP by having the verifier read the prover's messages in their entirety.

[4] It is sufficient to assume that $\beta = \frac{1}{2} \cdot \left(\frac{2 \cdot e \cdot \mathsf{l} \cdot \log \lambda}{c \cdot n}\right)^{-\mathsf{q}}$ to find contradiction in $\beta \leq \frac{1}{2} \cdot \left(\frac{2 \cdot e \cdot \mathsf{l} \cdot \log \lambda}{c \cdot n}\right)^{-\mathsf{q}}$ since we can always increase the soundness error without loss of generality.

Substituting the relevant parameters, we have that:

$$\begin{aligned} d &= \mathsf{l}' \cdot \log \lambda' + \mathsf{k}' \cdot \log \frac{\mathsf{k}'}{1-\alpha'-\beta'} \\ &= e \cdot \mathsf{l} \cdot (2\beta)^{1/\mathsf{q}} \cdot \log \lambda + \mathsf{k} \cdot \log(\mathsf{k}/\beta) \\ &= c \cdot n/2 + \mathrm{polylog}(n)\,. \end{aligned}$$

Thus, 3SAT is decidable in probabilistic time $2^{c \cdot n/2 + o(n)} < 2^{c \cdot n}$ in contradiction to RETH.

6.2.4.3 Limitations of high-round low-query IOPs

We sketch the proof of Theorem 13, showing that relations not decidable in few rounds do not have small-query IOPs with good soundness error. As in the theorem statement, let $\mathcal{R} \in \mathrm{AM}[\mathsf{k}] \backslash \mathrm{AM}[\mathsf{k}']$ be a relation for $\mathsf{k}' < \mathsf{k}$ and suppose that $\mathcal{R}$ has a k-round public-coin IOP $(\mathbf{P}, \mathbf{V})$ with perfect completeness, soundness error β, alphabet size $2^{\mathrm{poly}(|\mathbb{x}|)}$, proof length $\mathrm{poly}(|\mathbb{x}|)$, and query complexity $\mathsf{q} \leq \mathsf{k}'$.

By applying the round-reduction lemma (Item 2 of Lemma 5) to the k-round IOP $(\mathbf{P}, \mathbf{V})$ with parameter k', we get a k'-round IOP $(\mathbf{P}', \mathbf{V}')$ with completeness error $\alpha' := 1 - (\mathsf{k}'/(e \cdot \mathsf{k}))^{\mathsf{q}}$ and soundness error β. Suppose towards contradiction that $\beta < (\mathsf{k}'/(e \cdot \mathsf{k}))^{\mathsf{q}} - |\mathbb{x}|^{-c}$ for some $c \in \mathbb{N}$. Then the (additive) gap between completeness and soundness error of $(\mathbf{P}', \mathbf{V}')$ is $1 - \alpha' - \beta > |\mathbb{x}|^{-c}$.

Since the gap between completeness and soundness error of $(\mathbf{P}', \mathbf{V}')$ is inverse polynomial, it can be transformed into a k'-round public-coin IP $(\mathbf{P}''_{\mathsf{IP}}, \mathbf{V}''_{\mathsf{IP}})$ for $\mathcal{R}$ with completeness error $1/3$ and soundness error $1/3$. This is done by using the standard technique of taking $\mathrm{poly}(|\mathbb{x}|)$ parallel repetitions, computing the fraction of accepting transcripts, and accepting if the number of accepting transcripts is beyond some threshold that depends on α' and $|\mathbb{x}|^{-c}$. The IP $(\mathbf{P}''_{\mathsf{IP}}, \mathbf{V}''_{\mathsf{IP}})$, then, contradicts the assumption that $\mathcal{R} \notin \mathrm{AM}[\mathsf{k}']$.

6.2.4.4 Limitations of binary-alphabet constant-query IOPs

We sketch the proof of Theorem 14, showing that assuming RETH there are no binary-alphabet IOPs with 2 or 3 queries and small soundness error for 3SAT. We first discuss the following lemma which says that, assuming RETH, algorithms for solving constraint satisfaction problems (CSPs) cannot coexist with IOPs with a binary alphabet, constant query complexity, and small soundness error.

Lemma 8 (informal)**.** *Assume* RETH *and suppose that both of the following exist.*

- *An IOP with perfect completeness, soundness error β, round complexity k, alphabet size 2, proof length $2^{o(n)}$, query complexity q, verifier randomness r, and verifier running time $2^{o(n)}$.*
- *A polynomial-time algorithm $\mathbf{A}$ for deciding whether a binary-alphabet CSP with arity q has value 1 or value at most γ.*

Then $\beta > \gamma - \varepsilon$ *for every* ε *satisfying* $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$.

The proof of the theorem is concluded by relying on known algorithms for solving CSPs with appropriate arities q and decision bounds γ.

- For $\mathsf{q} = 2$, we rely on Schaefer's dichotomy theorem [Sch78], which says that the satisfiability of a binary-alphabet CSP with arity 2 can be decided in polynomial time. In this case $\gamma = 1$.
- For $\mathsf{q} = 3$, we rely on Zwick's algorithm [Zwi98], which decides in polynomial time whether a binary-alphabet CSP with arity 3 has value 1 or value smaller than $5/8$. In this case $\gamma = 5/8$.

Proof sketch of Lemma 8. Suppose towards contradiction that $\beta \leq \gamma - \varepsilon$ where ε satisfies $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$. The proof has two steps: (1) transform the IOP into a PCP for 3SAT that is "efficient-enough"; and (2) use the "efficient-enough" PCP and the algorithm $\mathbf{A}$ to decide 3SAT.

IOP to "efficient-enough" PCP. We apply these transformations from our toolbox.

1. **Derandomization using non-uniform advice.** Reduce the verifier randomness of the IOP using the non-uniform derandomization theorem (Lemma 7, Item 2) with error ε/k to get per-round randomness complexity of $O(\log(\mathsf{r} \cdot n/\varepsilon))$ bits. The new IOP uses $\mathrm{poly}(n, \mathsf{r}, 1/\varepsilon)$ bits of non-uniform advice, where a random string is good advice with overwhelming probability. The resulting IOP has perfect completeness, **soundness error** $\boldsymbol{\beta + \varepsilon \leq \gamma}$, round complexity k, alphabet size 2, proof length $2^{o(n)}$, query complexity q, **verifier randomness** $\boldsymbol{O(\log(\mathsf{r} \cdot n/\varepsilon))}$, and verifier running time $2^{o(n)} + \mathrm{poly}(n, \mathsf{r}, 1/\varepsilon) = 2^{o(n)}$.
2. **Unrolling to PCP.** Unroll the IOP into a PCP for 3SAT using Lemma 5, Item 3. This transformation preserves the number of advice bits, and also the fact that a random string is good advice with overwhelming probability. The resulting PCP has perfect completeness, soundness error γ, alphabet size 2, proof length $2^{O(\mathsf{k} \cdot \log(\mathsf{k} \cdot n/\varepsilon)) + o(n)} = 2^{o(n)}$, query complexity q, randomness complexity $O(\log(\mathsf{k} \cdot n/\varepsilon)) = o(n)$, and verifier running time $2^{o(n)}$.

Solving 3SAT using the PCP and CSP solvers. We use the PCP and the algorithm $\mathbf{A}$ to design a probabilistic algorithm $\mathbf{A}'$ that decides whether a 3SAT formula ϕ over n variables is satisfiable in time $2^{o(n)}$. The algorithm $\mathbf{A}'$, on input the 3SAT formula ϕ, works as follows.

1. **Sample random advice.** Sample a random advice string z for the PCP resulting from the previous transformation.
2. **Transform formula to CSP.** Transform the 3SAT formula ϕ into a binary-alphabet CSP ψ with arity q. This is done using the standard method of translating a PCP into a CSP; each constraint in the CSP is indexed by a choice of verifier randomness α and described by the verifier circuit with the input formula ϕ, randomness α, and advice z hard-coded. The CSP ψ has size $\mathrm{poly}(2^{\mathsf{r}'}, \mathsf{vt}') = 2^{o(n)}$ where $\mathsf{r}' = o(n)$ and

$\mathsf{vt}' = 2^{o(n)}$ are the randomness complexity and verifier running time of the PCP. Additionally, assuming that z is good advice, we have that if $\phi \in$ 3SAT then the value of ψ is 1, and if $\phi \notin$ 3SAT, then the value of ϕ is at most γ.

3. **Solve CSP.** Run $\mathbf{A}(\psi)$ and say that ϕ is satisfiable if and only if $\mathbf{A}$ says that ψ's value is 1.

The algorithm $\mathbf{A}'$ decides 3SAT with high probability: with overwhelming probability the choice of advice z is good, and deciding whether the value of the CSP instance ψ is 1 or γ, as $\mathbf{A}$ does, is equivalent to deciding whether ϕ is satisfiable.

Moreover, the algorithm $\mathbf{A}'$ runs in probabilistic time $2^{o(n)}$: the advice sampled in the first step is polynomial; the second step can be done in time $\mathrm{poly}(2^{\mathsf{r}'}, \mathsf{vt}') = 2^{o(n)}$ where $\mathsf{r}' = o(n)$ and $\mathsf{vt}' = 2^{o(n)}$ are the randomness complexity and verifier running time of the PCP; the final step takes $\mathrm{poly}(|\psi|) = 2^{o(n)}$, since $\mathbf{A}$ runs in polynomial time.

We obtained an algorithm for deciding 3SAT in probabilistic time $2^{o(n)}$, contradicting RETH. □

6.3 Tools for length and round reduction

We show several transformations for reducing the length of IOPs, either by reducing the number of rounds, or by reducing each round's proof length.

- Section 6.3.1: a length-reduction lemma for IOPs with small query complexity.
- Section 6.3.2: a round-reduction lemma for IOPs that do not query each of their rounds.
- Section 6.3.3: a statement that captures how IOPs can be "unrolled" into PCPs.

6.3.1 Length reduction

The lemma below shows that the length an IOP can be reduced, albeit with an increase in the completeness error.

Lemma 6.3.1. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ with total proof length $\mathsf{l}_{\mathsf{tot}}$ and query complexity q. Let ℓ be a parameter satisfying $\mathsf{q} \leq \ell \leq \mathsf{l}_{\mathsf{tot}}$. Then Theorem 6.3.2 yields a public-coin IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with the following parameters:*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (total)	$\mathsf{l}_{\mathsf{tot}}$
Queries	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

Round-reduced IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	$1 - (1-\alpha) \cdot \binom{\mathsf{l}_{\mathsf{tot}}-\mathsf{q}}{\ell-\mathsf{q}} / \binom{\mathsf{l}_{\mathsf{tot}}}{\ell} \leq 1 - (1-\alpha) \cdot (\ell / (e \cdot \mathsf{l}_{\mathsf{tot}}))^{\mathsf{q}}$
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (total)	ℓ
Queries	q
Randomness (per round)	$\mathsf{r} + \log \binom{\mathsf{l}_{\mathsf{tot}}}{\ell} \leq \mathsf{r} + \ell \cdot \log \mathsf{l}_{\mathsf{tot}}$
Verifier running time	$\mathrm{poly}(\mathsf{vt}, \ell, \log \mathsf{l}_{\mathsf{tot}})$

Moreover, if $(\mathbf{P}, \mathbf{V})$ is non-adaptive, then so is $(\mathbf{P}', \mathbf{V}')$.

Construction 6.3.2. The prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{V}'$: Sample a random set of ℓ indices $I := \{i_k\}_{k \in [\ell]} \in \binom{[\mathsf{l}_{\mathsf{tot}}]}{\ell}$ with $i_1 < \cdots < i_\ell$. Note that these indices span all of the prover messages. Send I to the prover.

2. For $j = 1, \ldots, \mathsf{k}$:
   (a) $\mathbf{V}'$: Randomly sample and send $\alpha_j \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) $\mathbf{P}'$: Compute $\Pi_j := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$. Send Π'_j equal to Π_j restricted to the indices appropriate to Π_j in I.
3. $\mathbf{V}'$: Given direct access to interaction randomness $\alpha_1, \ldots, \alpha_{\mathsf{k}}$ and oracle access to the prover messages $\Pi'_1, \ldots, \Pi'_{\mathsf{k}}$:
   (a) Simulate $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$ where, for any query made by the verifier to a location $j \in I$, the query is forwarded to the appropriate location proof oracle. If $\mathbf{V}$ queries some location $j \notin I$, then immediately reject.
   (b) Output the same answer as $\mathbf{V}$.

Proof of Theorem 6.3.1. We prove completeness, then soundness, and finally analyze complexity measures.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Consider the following algorithms $\mathbf{A}'$ and $\mathbf{A}$.

- $\mathbf{A}'(\mathbb{x}, \mathbb{w})$:
  1. Choose a random set $I \in \binom{[\ell_{\mathsf{tot}}]}{\ell}$.
  2. For $i \in [\mathsf{k}]$ run $\mathbf{V}$ to receive α_i and compute $\Pi_j := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$.
  3. Run $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$, where if $\mathbf{V}$ queries a location $j \in I$ then answer according to the appropriate proof. If $\mathbf{V}$ queries at a location $j \notin I$ then output 0 and quit.
  4. If the algorithm has not quit, output whatever $\mathbf{V}$ output.
- $\mathbf{A}(\mathbb{x}, \mathbb{w})$:
  1. Choose a random set $I \in \binom{[\ell_{\mathsf{tot}}]}{\ell}$.
  2. For $i \in [\mathsf{k}]$ run $\mathbf{V}$ to receive α_i and compute $\Pi_j := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_j)$.
  3. Run $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$. For every $j \in [\mathsf{k}]$, let Q be the set of locations queries by $\mathbf{V}$.
  4. If $Q \subseteq I$ then output whatever $\mathbf{V}$ output. Otherwise, output 0.

Notice that $\mathbf{A}'$ can be viewed as an emulation of $(\mathbf{P}', \mathbf{V}')$, and so:

$$\Pr\left[\langle \mathbf{P}'(\mathbb{w}), \mathbf{V}' \rangle(\mathbb{x}) = 1\right] = \Pr\left[\mathbf{A}'(\mathbb{x}, \mathbb{w}) = 1\right] .$$

Moreover, there is no difference between quitting whenever $\mathbf{V}$ queries outside of the set I and actually generating the result of $\mathbf{V}$ and only then quitting if $\mathbf{V}$ queried outside of I:

$$\Pr\left[\mathbf{A}'(\mathbb{x}, \mathbb{w}) = 1\right] = \Pr\left[\mathbf{A}(\mathbb{x}, \mathbb{w}) = 1\right] .$$

Finally, notice that the simulated prover and verifier in $\mathbf{A}(\mathbb{x}, \mathbb{w})$ constitute a random execution of $(\mathbf{P}(\mathbb{w}), \mathbf{V})$ on input $\mathbb{x}$. Fix Q of size q. The probability that I contains Q

is equal to $\binom{l_{tot}-q}{\ell-q}/\binom{l_{tot}}{\ell}$. Therefore:

$$\Pr\left[\mathbf{A}(\mathbb{x},\mathbb{w})=1\right] \geq \frac{\binom{l_{tot}-q}{\ell-q}}{\binom{l_{tot}}{\ell}} \cdot \Pr\left[\langle \mathbf{P}(\mathbb{w}),\mathbf{V}\rangle(\mathbb{x})=1\right].$$

Putting all of this together with the fact that the IOP $(\mathbf{P},\mathbf{V})$ has completeness error α, we have that

$$\begin{aligned}\Pr\left[\langle \mathbf{P}'(\mathbb{w}),\mathbf{V}'\rangle(\mathbb{x})=1\right] &\geq \frac{\binom{l_{tot}-q}{\ell-q}}{\binom{l_{tot}}{\ell}} \cdot \Pr\left[\langle \mathbf{P}(\mathbb{w}),\mathbf{V}\rangle(\mathbb{x})=1\right] \\ &\geq (1-\alpha)\cdot\frac{\binom{l_{tot}-q}{\ell-q}}{\binom{l_{tot}}{\ell}} \\ &> (1-\alpha)\cdot(\ell/(e\cdot l_{tot}))^q,\end{aligned}$$

where the final inequality follows from Theorem 2.7.3.

Soundness. Fix $\mathbb{x}\notin L(\mathcal{R})$ and a malicious prover $\tilde{\mathbf{P}}'$ for the IOP $(\mathbf{P}',\mathbf{V}')$. Consider the prover $\tilde{\mathbf{P}}$ for the IOP $(\mathbf{P},\mathbf{V})$ defined as follows:

- $\tilde{\mathbf{P}}$:
  1. Choose a random set $I\in\binom{[l_{tot}]}{\ell}$.
  2. For $j\in[k]$:
     (a) Receive α_j from the verifier $\mathbf{V}$.
     (b) Compute $\Pi'_j := \tilde{\mathbf{P}}'(\alpha_1,\ldots,\alpha_j)$ and let I_j be I restricted to the indices related to the j-th message of $\mathbf{P}$. Define Π_j as follows: for every $i\in I_j$, set $\mathbf{P}_j[i]$ to be equal to the appropriate location in Π'_j. For any $i\notin I_j$, set $\mathbf{P}_j[i]:=0$. Send Π_i to $\mathbf{V}$.

The sets I chosen by $\mathbf{V}'$ and $\tilde{\mathbf{P}}$ are chosen with the same probability. For every fixed $I=I^*$, the probability that $\mathbf{V}$ simulated by $\mathbf{V}'$ queries only in I is identical to the probability that the real $\mathbf{V}$ decides to query only in I in the interaction with $\tilde{\mathbf{P}}$. When there are only queries in I, the verifiers in both cases have an identical view, and so accept with the same probability. When the $\mathbf{V}$ simulated by $\mathbf{V}'$ queries outside of I, it immediately rejects, and, the real verifier $\mathbf{V}$ may or may not reject when it samples outside of I, since it reads from proofs padded by zeroes. Therefore

$$\Pr\left[\langle\tilde{\mathbf{P}}',\mathbf{V}'\rangle(\mathbb{x})=1\right] \leq \Pr\left[\langle\tilde{\mathbf{P}},\mathbf{V}\rangle(\mathbb{x})=1\right].$$

Finally, since the IOP $(\mathbf{P},\mathbf{V})$ has soundness error β, we conclude that

$$\Pr\left[\langle\tilde{\mathbf{P}}',\mathbf{V}'\rangle(\mathbb{x})=1\right] \leq \Pr\left[\langle\tilde{\mathbf{P}},\mathbf{V}\rangle(\mathbb{x})=1\right] \leq \beta.$$

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

- *Rounds.* The protocol has k rounds. (The message I and the first set of randomness sent by the verifier can be merged into a single message.)

- *Alphabet size and proof length.* The alphabet remains unchanged. The total proof length of the IOP is ℓ.
- *Queries.* The verifier makes q queries while simulating $\mathbf{V}$.
- *Randomness.* The verifier begins by sending $I \in \binom{[\mathsf{l}_{\mathsf{tot}}]}{\ell}$, which takes $\log \binom{\mathsf{l}_{\mathsf{tot}}}{\ell}$ bits. In the same round it sends its first random message of r bits. In each subsequent round, $\mathbf{V}'$ sends r bits of randomness. Thus the per-round interaction-randomness complexity is $\mathsf{r} + \log \binom{\mathsf{l}_{\mathsf{tot}}}{\ell} \leq \mathsf{r} + \ell \cdot \log \mathsf{l}_{\mathsf{tot}}$.
- *Verifier running time.* The running time of the verifier is $\mathrm{poly}(\mathsf{vt}, \ell, \log \mathsf{l}_{\mathsf{tot}})$.
- *Adaptivity.* The IOP is non-adaptive if the original IOP was non-adaptive.

□

6.3.2 Round reduction

The lemma below shows that the round complexity of an IOP can be reduced, albeit with an increase in the completeness error.

Lemma 6.3.3. *Let $\mathcal{R}$ be a relation with a public-coin k-round round-query IOP $(\mathbf{P}, \mathbf{V})$ with round-query complexity $\mathsf{q}_{\mathsf{round}}$. Let ℓ be a parameter such that $\mathsf{q}_{\mathsf{round}} \leq \ell \leq \mathsf{k}$. Then Theorem 6.3.4 yields a public-coin IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with parameters related*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Round queries	$\mathsf{q}_{\mathsf{round}}$
Queries (in total)	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

Round-reduced IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	$1 - (1-\alpha) \cdot \binom{\mathsf{k}-\mathsf{q}_{\mathsf{round}}}{\ell-\mathsf{q}_{\mathsf{round}}} / \binom{\mathsf{k}}{\ell} \leq 1 - (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k}))^{\mathsf{q}_{\mathsf{round}}}$
Soundness error	β
Rounds	$\ell + 1$
Alphabet size	λ
Proof length (per round)	l
Round queries	$\mathsf{q}_{\mathsf{round}}$
Queries (in total)	q
Randomness (per round)	$\mathsf{k} \cdot \mathsf{r} + \log \binom{\mathsf{k}}{\ell} \leq \mathsf{k} \cdot \mathsf{r} + \ell \cdot \log \mathsf{k}$
Verifier running time	$\mathrm{poly}(\mathsf{vt})$

Moreover, if $(\mathbf{P}, \mathbf{V})$ *is non-adaptive, then so is* $(\mathbf{P}', \mathbf{V}')$.

The goal of the construction is to decrease the number of rounds in the protocol to be equal to the number of prover messages that the verifier reads. The construction follows from the intuition that if the verifier does not read all of the prover's messages in an IOP, then the prover might as well not send them. Unfortunately, we cannot know the beginning of the protocol execution which of the prover's messages the verifier will read (in order to not send messages that will not be read). Instead, we have the verifier *guess* which of the prover's messages it will read and send this guess to the prover. The prover now skips over sending these messages.

If it turns out that the verifier chose correctly, then it has access to everything it needs to read. If it chose incorrectly, then it rejects. This behavior causes a degradation in completeness error, but the soundness error remains unchanged.

Construction 6.3.4. The prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{V}'$: Sample a random set of ℓ indices $I := \{i_j\}_{j \in [\ell]} \in \binom{[\mathsf{k}]}{\ell}$ with $i_1 < \cdots < i_\ell$. For notational convenience, set $i_0 := 0$. Send I to the prover.
2. For $j = 1, \ldots, \ell$:
   (a) $\mathbf{V}'$: Randomly sample and send $\alpha_{i_{(j-1)}+1}, \ldots, \alpha_{i_j} \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) $\mathbf{P}'$: Compute and send $\Pi_{i_j} := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i_j})$.
3. $\mathbf{V}'$: Given direct access to interaction randomness $\alpha_1, \ldots, \alpha_{i_\ell}$ and oracle access to the prover messages $\Pi_{i_1}, \ldots, \Pi_{i_\ell}$:
   (a) Randomly sample $\alpha_{(i_\ell+1)}, \ldots \alpha_{\mathsf{k}} \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) Simulate $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$ where if the verifier queries the proof Π_j for $j \in I$, the query is forwarded to the appropriate oracle. If $\mathbf{V}$ queries some $j \notin I$, then immediately reject.
   (c) Output the same answer as $\mathbf{V}$.

Remark 6.3.5. As written in Theorem 6.3.4, the protocol $(\mathbf{P}', \mathbf{V}')$ is not public-coin because the verifier's first message I dictates the length of subsequent verifier messages. Nevertheless, the protocol can be made public-coin by padding verifier messages to $\mathsf{k} \cdot \mathsf{r}$ bits. The prover and verifier act as in the protocol description, ignoring the padding bits.

Proof of Theorem 6.3.3. We analyze completeness, soundness, and complexity measures.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Consider the following algorithms $\mathbf{A}'$ and $\mathbf{A}$.

- $\mathbf{A}'(\mathbb{x}, \mathbb{w})$:
  1. Choose a random set $I \in \binom{[\mathsf{k}]}{\ell}$.

2. For $i \in [\mathsf{k}]$ run $\mathbf{V}$ to receive α_i and compute $\Pi_i := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_i)$.
3. Run $\mathbf{V}^{\Pi_1,\ldots,\Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$, where if $\mathbf{V}$ queries $j \in I$ then answer using Π_j. If $\mathbf{V}$ queries $j \notin I$ then output 0 and quit.
4. If the algorithm has not quit, output whatever $\mathbf{V}$ outputs.

- $\mathbf{A}(\mathbb{x}, \mathbb{w})$:
  1. Choose a random set $I \in \binom{[\mathsf{k}]}{\ell}$.
  2. For $i \in [\mathsf{k}]$ run $\mathbf{V}$ to receive α_i and compute $\Pi_i := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1, \ldots, \alpha_i)$.
  3. Run $\mathbf{V}^{\Pi_1,\ldots,\Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}})$. Let Q_{rnd} be the set of rounds queried by $\mathbf{V}$.
  4. If $Q_{\mathsf{rnd}} \subseteq I$ then output whatever $\mathbf{V}$ output. Otherwise output 0.

Notice that $\mathbf{A}'$ can be viewed as an emulation of $(\mathbf{P}', \mathbf{V}')$, and so:

$$\Pr\left[\langle \mathbf{P}'(\mathbb{w}), \mathbf{V}'\rangle(\mathbb{x}) = 1\right] = \Pr\left[\mathbf{A}'(\mathbb{x}, \mathbb{w}) = 1\right] .$$

Moreover, there is no difference between quitting whenever $\mathbf{V}$ queries outside of I and actually generating the result of $\mathbf{V}$ and only then quitting if $\mathbf{V}$ queried outside of I:

$$\Pr\left[\mathbf{A}'(\mathbb{x}, \mathbb{w}) = 1\right] = \Pr\left[\mathbf{A}(\mathbb{x}, \mathbb{w}) = 1\right] .$$

Finally, notice that the simulated prover and verifier in $\mathbf{A}(\mathbb{x}, \mathbb{w})$ constitute a random execution of $(\mathbf{P}(\mathbb{w}), \mathbf{V})$ on input $\mathbb{x}$. For a fixed Q_{rnd}, the probability that I contains Q_{rnd} is equal to $\binom{\mathsf{k}-\mathsf{q}_{\mathsf{round}}}{\ell-\mathsf{q}_{\mathsf{round}}}/\binom{\mathsf{k}}{\ell}$. Therefore:

$$\Pr\left[\mathbf{A}(\mathbb{x}, \mathbb{w}) = 1\right] > \frac{\binom{\mathsf{k}-\mathsf{q}_{\mathsf{round}}}{\ell-\mathsf{q}_{\mathsf{round}}}}{\binom{\mathsf{k}}{\ell}} \cdot \Pr\left[\langle \mathbf{P}(\mathbb{w}), \mathbf{V}\rangle(\mathbb{x}) = 1\right] .$$

Putting all of this together with the fact that the IOP $(\mathbf{P}, \mathbf{V})$ has completeness error α, we have that

$$\begin{aligned}
\Pr\left[\langle \mathbf{P}'(\mathbb{w}), \mathbf{V}'\rangle(\mathbb{x}) = 1\right] &> \frac{\binom{\mathsf{k}-\mathsf{q}_{\mathsf{round}}}{\ell-\mathsf{q}_{\mathsf{round}}}}{\binom{\mathsf{k}}{\ell}} \cdot \Pr\left[\langle \mathbf{P}(\mathbb{w}), \mathbf{V}\rangle(\mathbb{x}) = 1\right] \\
&\geq (1-\alpha) \cdot \frac{\binom{\mathsf{k}-\mathsf{q}_{\mathsf{round}}}{\ell-\mathsf{q}_{\mathsf{round}}}}{\binom{\mathsf{k}}{\ell}} \\
&> (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k}))^{\mathsf{q}_{\mathsf{round}}} ,
\end{aligned}$$

where the final inequality follows from Theorem 2.7.3.

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ and a malicious prover $\tilde{\mathbf{P}}'$ for the IOP $(\mathbf{P}', \mathbf{V}')$. Consider the prover for the IOP $(\mathbf{P}, \mathbf{V})$ defined as follows:

- $\tilde{\mathbf{P}}$:
  1. Choose a random set $I = \{i_j\}_{j \in [\ell]}$.
  2. For $i \in [\mathsf{k}]$:
     (a) Receive α_i from the verifier $\mathbf{V}$.

(b) If $i \in I$, compute $\Pi_i := \tilde{\mathbf{P}}'(\alpha_1, \ldots, \alpha_i)$. Otherwise, let Π_i be the all-zeroes proof. Send Π_i to $\mathbf{V}$.

The sets I chosen by $\mathbf{V}'$ and $\tilde{\mathbf{P}}$ are chosen with the same probability. For every fixed I, the probability that $\mathbf{V}$ simulated by $\mathbf{V}'$ queries only in I is identical to the probability that the real $\mathbf{V}$ decides to query only in I in the interaction with $\tilde{\mathbf{P}}$. When there are only queries in I, the verifiers in both cases have an identical view, and so accept with the same probability. When the $\mathbf{V}$ simulated by $\mathbf{V}'$ queries outside of I, it immediately rejects, and, the real verifier $\mathbf{V}$ may or may not reject when it samples outside of I, since it reads from the all-zeroes proofs. Therefore

$$\Pr\left[\langle \tilde{\mathbf{P}}', \mathbf{V}' \rangle(\mathbb{x}) = 1\right] \leq \Pr\left[\langle \tilde{\mathbf{P}}, \mathbf{V} \rangle(\mathbb{x}) = 1\right] .$$

Finally, since the IOP $(\mathbf{P}, \mathbf{V})$ has soundness error β, we conclude that

$$\Pr\left[\langle \tilde{\mathbf{P}}', \mathbf{V}' \rangle(\mathbb{x}) = 1\right] \leq \Pr\left[\langle \tilde{\mathbf{P}}, \mathbf{V} \rangle(\mathbb{x}) = 1\right] \leq \beta .$$

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

- *Rounds.* The protocol has $\ell + 1$ rounds. The message I and the first set of randomness sent by the verifier can be merged into a single message, but the final choice of randomness cannot be merged into a previous round.
- *Alphabet size and proof length.* The alphabet and proof length of the new IOP are identical to that of the original IOP.
- *Round queries.* The verifier reads queries from $\mathsf{q}_{\mathsf{round}}$ rounds.
- *Queries.* The verifier makes q queries while simulating $\mathbf{V}$.
- *Randomness.* The verifier first sends a random $I \in \binom{[\mathsf{k}]}{\ell}$, which takes $\log \binom{\mathsf{k}}{\ell}$ random bits. Following the padding of the interaction randomness (see Theorem 6.3.5), in each subsequent round $\mathbf{V}'$ sends $\mathsf{k} \cdot \mathsf{r}$ bits of randomness. Overall, the first round is the longest, in which the verifier sends $\mathsf{k} \cdot \mathsf{r} + \log \binom{\mathsf{k}}{\ell} \leq \mathsf{k} \cdot \mathsf{r} + \ell \cdot \log \mathsf{k}$ bits.
- *Verifier running time.* The running time of the verifier is $\mathrm{poly}(\mathsf{vt}, \mathsf{k}) = \mathrm{poly}(\mathsf{vt})$.
- *Adaptivity.* The IOP is non-adaptive if the original IOP was non-adaptive.

□

6.3.3 Unrolling IOPs to PCPs

The lemma below captures the folklore idea that any (public-coin) IOP can be unrolled into a PCP, whose proof length depends exponentially in the number of rounds and randomness.

Lemma 6.3.6. *Let $\mathcal{R}$ be a relation with a public-coin IOP system $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$. Then $\mathcal{R}$ has a PCP system $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ with parameters related as below.*

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt
Advice length	ℓ

$\longrightarrow$

PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Alphabet size	λ
Length	$\mathsf{l} \cdot 2^{O(\mathsf{k} \cdot \mathsf{r})}$
Queries	q
Randomness	$\mathsf{k} \cdot \mathsf{r}$
Verifier running time	$\mathrm{poly}(\mathsf{vt})$
Advice length	ℓ

If $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ is non-adaptive, then so is $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$. Moreover, any good advice string for $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ is also a good advice string for $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$.

Remark 6.3.7. A similar statement can be made for *private-coin* IOPs. For such IOPs the verifier may not sample random coins in each round, and as such we cannot reason about randomness *per round*, but rather consider the total number of random coins used by the verifier, $\mathsf{r}_{\mathsf{tot}}$. The PCP after unrolling has length $\mathsf{l} \cdot 2^{O(\mathsf{r}_{\mathsf{tot}})}$ and its verifier uses $\mathsf{r}_{\mathsf{tot}}$ random bits. A shorter length may be achieved by keeping track of the space of possible verifier messages rather than its randomness.

6.4 Tools for improving completeness

We show two transformations for reducing the completeness error of IOPs.

- Section 6.4.1: reducing the completeness error while increasing the alphabet size, and increasing the query complexity by 1.
- Section 6.4.2: achieving perfect completeness while increasing the alphabet size, and increasing the query complexity by 2.

6.4.1 Completeness amplification

The lemma below shows that the completeness error of an IOP can be reduced via a variation of parallel repetition, while the query complexity increases by 1.

Lemma 6.4.1. *Let $t \in \mathbb{N}$ be a parameter. Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$. Then Theorem 6.4.2 yields a public-coin IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with parameters related as below.*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

Completeness-amplified IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	α^t
Soundness error	$t \cdot \beta$
Rounds	$\mathsf{k} + 1$
Alphabet size	$\max\{\lambda, t\}$
Proof length (per round)	$t \cdot \mathsf{l}$
Queries	$\mathsf{q} + 1$
Randomness (per round)	$t \cdot \mathsf{r}$
Verifier running time	$\mathrm{poly}(\mathsf{vt}, t)$

Construction 6.4.2. The prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. $\mathbf{P}'$ and $\mathbf{V}'$ run t independent parallel executions of the interaction between $\mathbf{P}$ and $\mathbf{V}$ (excluding the decision stage) on instance $\mathbb{x}$ with witness $\mathbb{w}$. Let $\vec{\Pi}_i$ and $\vec{\alpha}_i$ be the proofs and interaction random strings corresponding to the i-th execution.
2. $\mathbf{P}'$:
   (a) If there exists an index $i \in [t]$, such that $\mathbf{V}^{\vec{\Pi}_i}(\mathbb{x}, \vec{\alpha}_i) = 1$, then send i to the verifier $\mathbf{V}'$ as a non-oracle message (choosing arbitrarily if there is more than one such index).
   (b) Otherwise, send an arbitrary value for i as a non-oracle message (or, alternatively, quit the execution).
3. $\mathbf{V}'$: Receive the value i (as a non-oracle message) from the prover. Check that $\mathbf{V}^{\vec{\Pi}_i}(\mathbb{x}, \vec{\alpha}_i)$=1, querying the appropriate proofs in the i-th execution as required by $\mathbf{V}$.

Notice that the verifier in this construction is adaptive.

Proof of Theorem 6.4.1. We analyze completeness, soundness, and complexity measures.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. By completeness of the original IOP, the verifier rejects in a parallel execution with probability at most α. Therefore, the probability that the verifier rejects in all executions is at most α^t. If there is any execution for which $\mathbf{V}$ accepts, then the prover sends its index as i, and $\mathbf{V}'$ accepts when emulating $\mathbf{V}$ on the i-th execution. We conclude that

$$\Pr\left[\langle \mathbf{P}'(\mathbb{w}), \mathbf{V}' \rangle(\mathbb{x}) = 0\right] \leq \alpha^t .$$

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ and a malicious prover $\tilde{\mathbf{P}}'$. By soundness of the original IOP, each of the parallel executions accepts with probability at most β. By applying the union bound over all t executions, we have that

$$\Pr\left[\langle \tilde{\mathbf{P}}', \mathbf{V}' \rangle(\mathbb{x}) = 1\right] \leq t \cdot \beta .$$

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

- *Rounds.* The protocol has $\mathsf{k} + 1$ rounds.
- *Alphabet size and proof length.* The alphabet of the new IOP is identical to that of the original IOP, with the addition of needing to send the index $i \in [t]$. Therefore the alphabet size is $\max\{\lambda, t\}$. The proof length is $t \cdot \mathsf{l}$, as it contains t executions of the original IOP.
- *Queries.* The verifier makes q queries while simulating $\mathbf{V}$, and one to read i. Therefore, the query complexity is $\mathsf{q} + 1$.
- *Randomness per-round.* In each round, $\mathbf{V}'$ sends $t \cdot \mathsf{r}$ bits of randomness corresponding to each parallel execution. Therefore the total randomness sent per round is $t \cdot \mathsf{r}$.
- *Verifier running time.* The running time of the verifier is at most $\mathrm{poly}(\mathsf{vt}, t)$.
- *Adaptivity.* The IOP is adaptive since the verifier's queries depend on the index i.

□

6.4.2 Perfect completeness

The lemma below shows that, for every IOP with good enough soundness error, perfect completeness can be achieved while preserving the query complexity up to constants.

Lemma 6.4.3. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$. Then Theorem 6.4.4 yields a public-coin IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with parameters related as below.*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

Perfectly-complete IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	0
Soundness error	$O\left(\beta \cdot \left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)\right)$
Rounds	$\mathsf{k}+1$
Alphabet size	$\max\{\lambda, 2^{\mathsf{r}\cdot\mathsf{k}}\}$
Proof length (per round)	$O\left(\mathsf{l} \cdot \left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)\right)$
Queries	$\mathsf{q}+2$
Randomness (per round)	r
Verifier running time	$\mathrm{poly}\left(\mathsf{vt}, \log\left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)\right)$

Moreover, if the verifier $\mathbf{V}$ *uses non-uniform advice, then* $\mathbf{V}'$ *uses advice of the same length.*

Construction 6.4.4. Let $t := 2 \cdot \left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)$. The prover $\mathbf{P}'$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, while the verifier $\mathbf{V}'$ receives as input the instance $\mathbb{x}$. They interact as follows.

- $\mathbf{P}'$: Send $\vec{z}_1, \ldots, \vec{z}_t = (z_{1,1}, \ldots, z_{1,\mathsf{k}}), \ldots, (z_{t,1}, \ldots z_{t,\mathsf{k}}) \in \{0,1\}^{\mathsf{r}\cdot\mathsf{k}}$ to the verifier.
- For $i = 1, \ldots, \mathsf{k}$:
  - $\mathbf{V}'$: Choose $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}}$ uniformly and send to the prover.
  - $\mathbf{P}'$: Send $\{\Pi_{j,i}\}_{j\in[t]}$ where $\Pi_{j,i} := \mathbf{P}(\mathbb{x}, \mathbb{w}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha_i \oplus z_{j,i})$.
- $\mathbf{P}'$:
  1. If there exists an index $j \in [t]$, such that $\mathbf{V}^{\Pi_{j,1},\ldots,\Pi_{j,\mathsf{k}}}(\mathbb{x}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha \oplus z_{j,\mathsf{k}}) = 1$, then send j to the verifier $\mathbf{V}'$ as a non-oracle message (choosing arbitrarily if there is more than one possible value for j).
  2. Otherwise, send $\perp$.
- $\mathbf{V}'$: Receive j as a non-oracle message.
  1. If $j = \perp$, then reject.
  2. Otherwise, query $\vec{z}_j = (z_{j,1}, \ldots, z_{j,\mathsf{k}})$ and check that $\mathbf{V}^{\Pi_{j,1},\ldots,\Pi_{j,\mathsf{k}}}(\mathbb{x}, \alpha_1 \oplus z_{j,1}, \ldots, \alpha_\mathsf{k} \oplus z_{j,\mathsf{k}}) = 1$, querying the appropriate proofs as required by $\mathbf{V}$.

While this construction (and subsequent analysis) assumes that $\mathbf{V}$ uses no non-uniform advice, the non-uniform case is identical, except that whenever any party invokes $\mathbf{V}$, it also passes along the advice string. Notice that the verifier in this construction is adaptive.

Proof of Theorem 6.4.3. We analyze completeness, soundness, and complexity measures. Completeness and soundness are identical to those in [FGMSZ89], and are repeated here for convenience.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. For convenience of notation, we denote $\vec{\alpha} := (\alpha_1, \ldots, \alpha_\mathsf{k})$. Thus, $\vec{\alpha} \oplus \vec{z}_j = (\alpha_1 \oplus z_{j,1}, \ldots, \alpha_\mathsf{k} \oplus z_{j,\mathsf{k}})$ for any j. By completeness of the original IOP, the verifier rejects in a parallel execution with probability at most α. We show the following claim:

Claim 6.4.5. *There exist strings* $\vec{z}_1, \ldots, \vec{z}_t = (z_{1,1}, \ldots, z_{1,k}), \ldots, (z_{t,1}, \ldots z_{t,k})$ *such that for every choice of verifier messages* $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$ *there is some* $j \in [t]$ *satisfying*

$$\mathbf{V}^{\Pi_{j,1},\ldots,\Pi_{j,k}}(\mathbb{x}, \vec{\alpha} \oplus \vec{z}_j) = 1\,,$$

where the prover messages $\Pi_{j,i}$ *are generated as in Theorem 6.4.4.*

This fact completes the proof, as the honest prover will be able to send these values $\vec{z}_1, \ldots, \vec{z}_t$, and, following the interaction with the verifier, is able to send the correct j to the verifier.

Proof of Theorem 6.4.5. Let S be the set of verifier randomness $\vec{\alpha}$ for which the original prover $\mathbf{P}$ is able to convince the verifier $\mathbf{V}$ on instance $\mathbb{x}$ and witness $\mathbb{w}$. By completeness of the protocol:

$$\Pr_{\vec{z}}[\vec{z} \notin S] < \alpha\,.$$

We say that a sequence $\vec{z}_1, \ldots, \vec{z}_t$ is *good* if for every $\vec{\alpha}$, there exists $j \in [t]$ with $\vec{\alpha} \oplus \vec{z}_j \in S$. We show that the probability that a uniformly random $\vec{z}_1, \ldots, \vec{z}_t$ is good is non-zero, which implies that the prover $\mathbf{P}'$ can always find such a sequence and manages to convince the verifier $\mathbf{V}'$:

$$\begin{aligned}
&\Pr_{\vec{z}_1,\ldots,\vec{z}_t}\left[\ \vec{z}_1, \ldots, \vec{z}_t \text{ are not good }\right] \\
&= \Pr_{\vec{z}_1,\ldots,\vec{z}_t}\left[\ \exists \vec{\alpha}\ \ \forall j \in [t]\ \vec{\alpha} \oplus \vec{z}_j \notin S\ \right] \\
&\leq 2^{r\cdot k} \cdot \Pr_{\vec{z}_1,\ldots,\vec{z}_t}\left[\ \forall j \in [t]\ \vec{z}_j \notin S\ \right] && (6.1) \\
&< 2^{r\cdot k} \cdot \alpha^t \\
&< 1\,, && (6.2)
\end{aligned}$$

where Equation 6.1 holds since $\vec{\alpha}$ is $r \cdot k$ bits long, and by the fact that for every fixed string $\vec{\alpha}$, the probability that $\vec{\alpha} \oplus \vec{z}_j \in S$ for a uniformly random $\vec{z}_j$ is equal to the probability that $\vec{z}_j \in S$. Equation 6.2 holds since $t > \frac{r\cdot k}{\log(1/\alpha)}$. □

Soundness. Fix $\mathbb{x} \notin L(\mathcal{R})$ and a malicious prover $\tilde{\mathbf{P}}'$. Suppose towards contradiction that $\tilde{\mathbf{P}}'$ is able to convince $\mathbf{V}'$ with probability greater than

$$\beta' := \beta \cdot t = 2 \cdot \beta \cdot \left(\frac{r \cdot k}{\log(1/\alpha)}\right)\,.$$

We construct a prover $\tilde{\mathbf{P}}$ that manages to convince $\mathbf{V}$ to accept on $\mathbb{x}$ with probability greater than β, which contradicts the soundness error of the original IOP.

$\tilde{\mathbf{P}}$ begins by using $\tilde{\mathbf{P}}'$ on $\mathbb{x}$ to generate a sequence of strings

$$\vec{z}_1, \ldots, \vec{z}_t = (z_{1,1}, \ldots, z_{1,k}), \ldots, (z_{t,1}, \ldots z_{t,k})\,.$$

It then chooses a random $j \in [t]$. During interaction with the verifier, upon receiving a string α_i, $\tilde{\mathbf{P}}$ passes $\alpha'_i := \alpha_i \oplus z_{i,j}$ to $\tilde{\mathbf{P}}'$ and receives proofs $\Pi_{1,i}, \ldots, \Pi_{t,i}$.

Whenever $\tilde{\mathbf{P}}'$ convinces $\mathbf{V}'$ to accept by telling $\mathbf{V}'$ to read the shift string $\vec{z}_j$, this means that $\mathbf{V}$ accepts given proofs $\Pi_{j,1}, \ldots, \Pi_{j,\mathsf{k}}$ and randomness $(\alpha'_1 \oplus z_{j,1}), \ldots, (\alpha'_{\mathsf{k}} \oplus z_{j,\mathsf{k}})$. Since $\alpha'_i := \alpha_i \oplus z_{i,j}$, this means that $\mathbf{V}$ accepts given proofs $\Pi_{j,1}, \ldots, \Pi_{j,\mathsf{k}}$ and randomness $\alpha_1, \ldots, \alpha_{\mathsf{k}}$. Hence, $\tilde{\mathbf{P}}$ is able to cause the verifier to accept whenever $\tilde{\mathbf{P}}'$ manages to do so with the same index j chosen by $\tilde{\mathbf{P}}$. Therefore, the probability that $\tilde{\mathbf{P}}$ causes the verifier to accept is greater than $\beta'/t = \beta$, which contradicts the soundness error of the original IOP $(\mathbf{P}, \mathbf{V})$.

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

- *Rounds.* The protocol has $\mathsf{k}+1$ rounds as the prover's final message can be merged with its previous message.
- *Alphabet size and proof length.* The alphabet of the new IOP is identical to that of the original IOP, with the addition of needing to send the strings $\vec{z}_1, \ldots, \vec{z}_t$ (which are each read as a single block by the verifier) and the index $j \in [t] \cup \{\perp\}$. Therefore the alphabet size is $\max\{\lambda, 2^{\mathsf{r}\cdot\mathsf{k}}\}$. The proof length of the first prover message (the vectors $\vec{z}$) is t, and the length of the rest of the messages is $\mathsf{l} \cdot t$ since the prover sends t different proofs of the original IOP to the verifier. The proof length is, therefore, $\mathsf{l} \cdot t = 2 \cdot \mathsf{l} \cdot \left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)$ symbols.
- *Queries.* The verifier makes one query to read j, one query to read $\vec{z}_j$ and q queries while simulating $\mathbf{V}$. Therefore, the query complexity is $\mathsf{q}+2$.
- *Randomness per-round.* The randomness complexity of the IOP is r.
- *Verifier running time.* The running time of the verifier is $\mathrm{poly}(\mathsf{vt}, \log t) = \mathrm{poly}\left(\mathsf{vt}, \log\left(\frac{\mathsf{r}\cdot\mathsf{k}}{\log(1/\alpha)}\right)\right)$.
- *Adaptivity.* The IOP is adaptive since the verifier's queries depend of the index j.

□

6.5 Tools for derandomization

We show how to derandomize a public-coin IOP based on non-uniform advice (Section 6.5.1) or based on pseudorandom generators (Section 6.5.2). In both cases, the verifier's randomness is reduced by having the IOP verifier sample each round's randomness from a smaller set; in Section 6.5.1 this set is the non-uniform advice, and in Section 6.5.2 this set is the PRG's seeds.

6.5.1 Derandomization using non-uniform advice

Theorem 6.5.1. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ and $\varepsilon = \varepsilon(|\mathbb{x}|) \in (0,1]$ be a noticeable function. Then Theorem 6.5.2 is a public-coin IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with parameters related as below.*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

Non-uniform IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	$\alpha + \mathsf{k} \cdot \varepsilon$
Soundness error	$\beta + \mathsf{k} \cdot \varepsilon$
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	$\Theta\left(\log\left((\mathsf{r} \cdot \mathsf{k} + \lvert\mathbb{x}\rvert)/\varepsilon\right)\right)$
Verifier running time	$O(\mathsf{vt} + \mathsf{r} \cdot (\mathsf{r} \cdot \mathsf{k} + \lvert\mathbb{x}\rvert)/\varepsilon^2)$
Advice length	$\Theta(\mathsf{r} \cdot (\mathsf{r} \cdot \mathsf{k} + \lvert\mathbb{x}\rvert)/\varepsilon^2)$

Moreover, if $\alpha = 0$ then $(\mathbf{P}', \mathbf{V}')$ has completeness error 0. Finally, for every instance $\mathbb{x}$, a uniformly random string of length $\Theta(\mathsf{r} \cdot (\mathsf{r} \cdot \mathsf{k} + |\mathbb{x}|)/\varepsilon^2)$ is good advice with probability at least $1 - 2^{-|\mathbb{x}|}$.

Construction 6.5.2. On instance $\mathbb{x}$, witness $\mathbb{w}$, and non-uniform advice $Y \subseteq \{0,1\}^{\mathsf{r}}$ where $|Y| = \Theta((\mathsf{r} \cdot \mathsf{k} + |\mathbb{x}|)/\varepsilon^2)$, the IOP $(\mathbf{P}', \mathbf{V}')$ proceeds as follows.

1. For $j = 1, \ldots, \mathsf{k}$:
   (a) $\mathbf{V}'$: Sample a random string $\alpha_j \leftarrow Y$ and send it to the prover. This involves sampling and sending $O(\log |Y|)$ bits.
   (b) $\mathbf{P}'$: Given verifier messages $\alpha_1, \ldots, \alpha_j$, send a message Π_j.
2. $\mathbf{V}'$: Given oracle access to $\Pi_1, \ldots, \Pi_{\mathsf{k}}$, accept if and only if

$$\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}}) = 1\,.$$

In order to prove Theorem 6.5.1, we show a generic lemma that bounds the effect of the sub-sampling on the average value of the interaction tree of a the protocol.

Lemma 6.5.3. *Let $T_{\mathbb{x}}$ be an interaction tree of IOP $(\mathbf{P},\mathbf{V})$ on input $\mathbb{x}$. For a set $Y \subseteq \{0,1\}^{\mathsf{r}}$ of size $t = \Theta((\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon^2)$, let $T_{\mathbb{x},Y}$ be the sub-tree of $T_{\mathbb{x}}$ where all verifier messages are chosen only from Y. Then*

$$\Pr_{Y}\left[|\mathsf{val}(T_{\mathbb{x},Y}) - \mathsf{val}(T_{\mathbb{x}})| \geq \mathsf{k}\cdot\varepsilon\right] \leq 1 - 2^{-|\mathbb{x}|}/3\,.$$

In particular (by applying the union bound), there exists a set Y such that for every $\mathbb{x}$:

$$|\mathsf{val}(T_{\mathbb{x},Y}) - \mathsf{val}(T_{\mathbb{x}})| \geq \mathsf{k}\cdot\varepsilon\,.$$

First, we use Theorem 6.5.3 to prove Theorem 6.5.1, and then prove Theorem 6.5.3.

Proof of Theorem 6.5.1. We analyze completeness, soundness, and the efficiency parameters.

Completeness. Fix $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$. Let $T_{\mathbb{x}}$ be the interaction tree of $(\mathbf{P},\mathbf{V})$ on input $\mathbb{x}$ and $T_{\mathbb{x},\mathsf{G}}$ be the interaction tree of $(\mathbf{P}',\mathbf{V}')$ on input $\mathbb{x}$ and given PRG G. We have $\mathsf{val}(T_{\mathbb{x}}) \geq 1-\alpha$. By Theorem 6.5.3:

$$|\mathsf{val}(T_{\mathbb{x}}) - \mathsf{val}(T_{\mathbb{x},\mathsf{G}})| \leq \mathsf{k}\cdot\varepsilon\,,$$

and so $\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \geq 1-(\alpha+\mathsf{k}\cdot\varepsilon)$. It follows that there exists a prover strategy that causes the $\mathbf{V}'$ to accept with probability at least $1-(\alpha+\mathsf{k}\cdot\varepsilon)$.

Soundness. Fix some instance $\mathbb{x} \notin L(\mathcal{R})$. Let $T_{\mathbb{x}}$ be the interaction tree of $(\mathbf{P},\mathbf{V})$ on input $\mathbb{x}$ and $T_{\mathbb{x},\mathsf{G}}$ be the interaction tree of $(\mathbf{P}',\mathbf{V}')$ on input $\mathbb{x}$ and given PRG G. We have $\mathsf{val}(T_{\mathbb{x}}) \leq \beta$. By Theorem 6.5.3:

$$|\mathsf{val}(T_{\mathbb{x}}) - \mathsf{val}(T_{\mathbb{x},\mathsf{G}})| \leq \mathsf{k}\cdot\varepsilon\,,$$

and so $\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \leq \beta+\mathsf{k}\cdot\varepsilon$. It follows that there no malicious prover strategy can cause $\mathbf{V}'$ to accept with probability greater than $\beta+\mathsf{k}\cdot\varepsilon$.

Complexity measures. We analyze the efficiency parameters of the resulting IOP:

- *Rounds.* The protocol has k rounds, as in the original IOP $(\mathbf{P},\mathbf{V})$.
- *Alphabet size and proof length.* The alphabet and proof length of the new IOP are identical to that of the original IOP.
- *Advice length.* The advice $Y \subseteq \{0,1\}^{\mathsf{r}}$ contains $\Theta((\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon^2)$ strings of length r. Therefore the total length of the advice is $\Theta(\mathsf{r}\cdot(\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon^2)$.
- *Queries.* The IOP verifier reads at most q symbols, as in the original IOP $(\mathbf{P},\mathbf{V})$.
- *Randomness per-round.* The verifier, in each round, samples a random element from Y. Since $|Y| = \Theta((\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon^2)$, the randomness complexity is $\Theta(\log\left(\Theta((\mathsf{r}\cdot\mathsf{k}+|\mathbb{x}|)/\varepsilon^2)\right))$.

- *Verifier running time.* The verifier runs in time $O(\mathsf{vt} + \mathsf{r} \cdot (\mathsf{r} \cdot \mathsf{k} + |\mathbb{x}|)/\varepsilon^2)$ since it must read all of its advice.
- *Adaptivity.* The IOP is non-adaptive if the original IOP was non-adaptive.

□

Proof of Theorem 6.5.3. First, we prove that, for every node on the tree, if in this node the verifier samples its messages from the set Y (rather than $\{0,1\}^{\mathsf{r}}$) then with high probability (over the choice of Y) the value of the subtree following the node does not change by much.

Claim 6.5.4. *Let T be an interaction (sub-)tree whose root corresponds to the verifier's turn to send a message, and, for a set $Y \subseteq \{0,1\}^{\mathsf{r}}$ of size t, let T_Y be the tree T when the verifier message corresponding to the root is chosen only from Y. Then:*

$$\Pr_Y \left[|\mathsf{val}(T_Y) - \mathsf{val}(T)| \geq \varepsilon\right] \leq 2 \cdot \exp(-2\varepsilon^2 \cdot t)\,.$$

Proof. For $\alpha \in \{0,1\}^{\mathsf{r}}$, let $T(\alpha)$ be the subtree of T when α is chosen as the verifier's message. Then by definition:

$$\mathsf{val}(T) := \mathbb{E}_{\alpha \leftarrow \{0,1\}^{\mathsf{r}}}[\mathsf{val}(T(\alpha))]\,.$$

Similarly, for every Y: $\mathsf{val}(T_Y) = \mathbb{E}_{\alpha \leftarrow Y}[\mathsf{val}(T(\alpha))]$. Notice that for every Y, Y' of size t that are identical apart for only one string,

$$|\mathsf{val}(T_Y) - \mathsf{val}(T_{Y'})| \leq 1/t\,,$$

since the largest difference occurs if $\mathsf{val}(T(\alpha)) = 1$ and $\mathsf{val}(T(\alpha')) = 0$, where α and α' are the elements that differ between the sets. We can therefore apply Theorem 2.7.2 with $f(Y) := \mathsf{val}(T_Y)$ and value $\sigma_j = 1/t$ for every j, noting that $\mathsf{val}(T) := \mathbb{E}_{\alpha \leftarrow \{0,1\}^{\mathsf{r}}}[\mathsf{val}(T(\alpha))] = \mathbb{E}_Y[f(Y)]$ to get

$$\Pr_Y[|\mathsf{val}(T_Y) - \mathsf{val}(T)| \geq \varepsilon] \leq 2 \cdot \exp(-2\varepsilon^2 \cdot t)\,.$$

□

We now apply Theorem 6.5.4 to the entire tree $T_{\mathbb{x}}$ to turn it into the tree $T_{\mathbb{x},Y}$, showing that with high probability that the value of the tree does not change by much. We do this in a layer-by-layer fashion from the bottom up. For $i \in \{0,\ldots,\mathsf{k}\}$, let T_i be the tree $T_{\mathbb{x}}$ where nodes in layers $1,\ldots,i$ have the verifier sampling uniformly, and nodes in layers $i+1,\ldots,\mathsf{k}$ are changed to subsample from Y.

Claim 6.5.5. *For every $i \in [\mathsf{k}]$:*

$$\Pr_Y[|\mathsf{val}(T_{i-1}) - \mathsf{val}(T_i)| \geq \varepsilon] \leq 2 \cdot \ell_i \cdot \exp(-2\varepsilon^2 \cdot t)\,,$$

where ℓ_i is the number of nodes in layer i of T for which the next message is a verifier message.

Proof. By Theorem 6.5.4, switching a single node changes the value of its subtree by a value more than ε with probability at most $2 \cdot \exp(-2\varepsilon^2 \cdot t)$. By applying the union bound over all ℓ_i nodes in layer i, we have that there is no node in layer i whose subtree changes value by more than ε with probability at least $2 \cdot \ell_i \cdot \exp(-2\varepsilon^2 \cdot t)$.

The claim follows by noticing that since all of the nodes that have been changed are in the same layer, the expected value of the tree also has a change of at most ε. □

By applying Theorem 6.5.5 and noticing that $T_0 = T_{\mathbb{x},Y}$ and $T_{\mathsf{k}} = T_{\mathbb{x}}$, we have that:

$$\Pr_Y[|\mathsf{val}(T_{\mathbb{x}}) - \mathsf{val}(T_{\mathbb{x},Y})| \geq \mathsf{k} \cdot \varepsilon] \leq \Pr_Y[\exists i \text{ s.t. } |\mathsf{val}(T_{i-1}) - \mathsf{val}(T_i)| \geq \varepsilon] \tag{6.3}$$

$$\leq 2 \cdot \exp(-2\varepsilon^2 \cdot t) \cdot \sum_i \ell_i \tag{6.4}$$

$$\leq 2^{\mathsf{k} \cdot \mathsf{r}+1} \cdot \exp(-2\varepsilon^2 \cdot t)\,. \tag{6.5}$$

Equation 6.3 can be verified by a simple counting argument, since there are k layers. Equation 6.4 follows by applying Theorem 6.5.5 and the union bound, and Equation 6.5 follows from the fact that since in every one of the k rounds the verifier sends r random bits, and so the tree T contains at most $2^{\mathsf{k} \cdot \mathsf{r}}$ nodes.

By setting $t = \Theta((\mathsf{r} \cdot \mathsf{k} + |\mathbb{x}|)/\varepsilon^2)$, we have that:

$$\Pr_Y\left[|\mathsf{val}(T_{\mathbb{x},Y}) - \mathsf{val}(T_{\mathbb{x}})| < \mathsf{k} \cdot \varepsilon\right] < 1 - 2^{-|\mathbb{x}|}/3\,.$$

□

6.5.2 Derandomization using pseudorandom generators

Theorem 6.5.6. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}, \mathbf{V})$ with (per round) randomness complexity r, and communication complexity l. Let $\mathsf{G}\colon \{0,1\}^{\ell_{\mathsf{PRG}}} \to \{0,1\}^{\mathsf{r}}$ be a PRG against circuits of size $\mathsf{s}_{\mathsf{PRG}} = \mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ with* PSPACE *gates. Then Theorem 6.5.7 yields an IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$ with parameters:*

IOP $(\mathbf{P}, \mathbf{V})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	$\mathsf{vt} = \mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \mathsf{l}, \mathsf{r})$

+

PRG G	
Seed length	ℓ_{PRG}
Error	ϵ_{PRG}
Security against size	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \mathsf{l}, \mathsf{r})$
Running time	$\mathsf{t}_{\mathsf{PRG}}$

IOP $(\mathbf{P}', \mathbf{V}')$ for $\mathcal{R}$	
Completeness error	$1 - ((1-\alpha)/3 + 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2)$
Soundness error	$3\beta + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3$
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	ℓ_{PRG}
Verifier running time	$O(\mathsf{vt} + \mathsf{k} \cdot \mathsf{t}_{\mathsf{PRG}})$

$\longrightarrow$

Moreover, if $\alpha = 0$ then $(\mathbf{P}', \mathbf{V}')$ has completeness error 0.

Construction 6.5.7. Suppose without loss of generality that the honest prover $\mathbf{P}$ maximizes the probability that $\mathbf{V}$ accepts given $\mathbb{x} \in L(\mathcal{R})$. On instance $\mathbb{x}$ and witness $\mathbb{w}$, the protocol $(\mathbf{P}', \mathbf{V}')$ proceeds as follows:

1. For $j = 1, \ldots, \mathsf{k}$:
   (a) $\mathbf{V}'$: Choose a uniformly random string $\alpha_j \leftarrow \{0,1\}^{\ell_{\mathsf{PRG}}}$ and send it to the prover.
   (b) $\mathbf{P}'$: Send some message Π_j to the verifier.
2. $\mathbf{V}'$: Given oracle access to $\Pi_1, \ldots, \Pi_{\mathsf{k}}$, accept if and only if $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \mathsf{G}(\alpha_1), \ldots, \mathsf{G}(\alpha_{\mathsf{k}})) = 1$, querying the oracles $\Pi_1, \ldots, \Pi_{\mathsf{k}}$ appropriately.

In order to prove Theorem 6.5.6, we show a generic lemma that bounds the effect of the sub-sampling on the average value of the interaction tree of a the protocol.

Lemma 6.5.8. *Let $T_{\mathbb{x}}$ be an interaction tree of IOP $(\mathbf{P}, \mathbf{V})$ on input $\mathbb{x}$. For a PRG G against circuits of size $\mathsf{s}_{\mathsf{PRG}} = \mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ with* PSPACE *gates, let $T_{\mathbb{x},\mathsf{G}}$ be the tree $T_{\mathbb{x}}$ when the verifier messages are chosen only using G. Then*

$$\mathsf{val}(T_{\mathbb{x}})/3 - 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2 \leq \mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \leq 3\mathsf{val}(T_{\mathbb{x}}) + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3 .$$

We first use Theorem 6.5.8 to prove Theorem 6.5.6, and later give a proof of Theorem 6.5.8.

Proof of Theorem 6.5.6. We analyze completeness, then soundness, and finally the efficiency parameters of the resulting IOP.

Completeness. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Let $T_{\mathbb{x}}$ be the interaction tree of $(\mathbf{P}, \mathbf{V})$ on input $\mathbb{x}$ and $T_{\mathbb{x},\mathsf{G}}$ be the interaction tree of $(\mathbf{P}', \mathbf{V}')$ on input $\mathbb{x}$ and given PRG G. We have $\mathsf{val}(T_{\mathbb{x}}) \geq 1 - \alpha$. By Theorem 6.5.8:

$$\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \geq \mathsf{val}(T_{\mathbb{x}})/3 - 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2 ,$$

and so $\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \geq (1-\alpha)/3 + 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2$. It follows that there exists a prover strategy that causes the $\mathbf{V}'$ to accept with probability at least $(1-\alpha)/3 + 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2$.

Soundness. Fix some instance $\mathbb{x} \notin L(\mathcal{R})$. Let $T_{\mathbb{x}}$ be the interaction tree of $(\mathbf{P}, \mathbf{V})$ on input $\mathbb{x}$ and $T_{\mathbb{x},\mathsf{G}}$ be the interaction tree of $(\mathbf{P}', \mathbf{V}')$ on input $\mathbb{x}$ and given PRG G. We have $\mathsf{val}(T_{\mathbb{x}}) \leq \beta$. By Theorem 6.5.8:

$$\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \leq 3\mathsf{val}(T_{\mathbb{x}}) + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3 ,$$

and so $\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \leq 3\beta + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3$. It follows that there no malicious prover strategy can cause $\mathbf{V}'$ to accept with probability greater than $3\beta + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3$.

Complexity measures. We analyze the efficiency parameters of the resulting IOP:

- *Rounds.* The protocol has k rounds, as in the original IOP $(\mathbf{P}, \mathbf{V})$.
- *Alphabet size and proof length.* The alphabet and proof length of the new IOP are identical to that of the original IOP.
- *Queries.* The IOP verifier reads at most q symbols, as in the original IOP $(\mathbf{P}, \mathbf{V})$.
- *Randomness per-round.* The verifier, in each round, samples a random seed for G of length ℓ_{PRG}.
- *Verifier running time.* The verifier runs in time $O(\mathsf{vt} + \mathsf{k} \cdot \mathsf{t}_{\mathsf{PRG}})$.
- *Adaptivity.* The IOP is non-adaptive if the original IOP was non-adaptive.

□

Proof of Theorem 6.5.8. First, we show that, for every node in the tree, if in this node the verifier samples its messages using the PRG then the value of the subtree following the node does not change by much.

Claim 6.5.9. *Let T' be an interaction (sub-)tree corresponding to an instance $\mathbb{x}$ and transcript prefix tr whose root corresponds to the verifier's turn to send a message. Furthermore, let T'_{G} be the tree T' when the root verifier message is chosen only using the PRG G. Then:*

$$(1 + 1/\mathsf{k})^{-1} \cdot \left(\mathsf{val}(T') - 12\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}\right) \leq \mathsf{val}(T'_{\mathsf{G}}) \leq (1 + 1/\mathsf{k}) \cdot \left(\mathsf{val}(T') + 18 \cdot \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2\right) .$$

Proof. For the verifier's message α denote $T'(\alpha)$ be the sub-tree of T' where α is chosen as the verifier's first message. Then by definition: $\mathbb{E}_{\alpha}[\mathsf{val}(T'(\alpha))] = \mathsf{val}(T')$.

Let $\delta := \frac{1}{3\mathsf{k}}$. We define ranges of values for $\mathsf{val}(T'(\alpha))$. Let $m \in \mathbb{N}$ be the minimal integer such that $(1+\delta)^m \cdot \delta \cdot \mathsf{val}(T') \geq 1$. For every $i \in \{0, 1, \ldots, m\}$ let $R_i = (1+\delta)^{-i}$, and set $R_{m+1} = 0$. For every $i \in \{0, 1, \ldots, m\}$, let $p_i = \Pr_{\alpha}[R_i \leq \mathsf{val}(T'(\alpha)) \leq R_{i-1}]$. Notice that:

$$\sum_{i=1}^{m+1} p_i \cdot R_i \leq \mathsf{val}(T') \leq \sum_{i=1}^{m+1} p_i \cdot R_{i-1} .$$

Additionally, notice that since $R_{i-1} = (1+\delta) \cdot R_i$ for every $i \neq m+1$:

$$\sum_{i=1}^{m+1} p_i \cdot R_i = \sum_{i=1}^{m} p_i \cdot R_i = (1+\delta)^{-1} \cdot \sum_{i=1}^{m} p_i \cdot R_{i-1} \,.$$

By noting that $\sum_{i=1}^{m+1} p_i \cdot R_{i-1} \leq (1+\delta)^{-m} + \sum_{i=1}^{m} p_i \cdot R_{i-1}$, we conclude the following bounds on $\mathsf{val}(T')$:

$$(1+\delta)^{-1} \cdot \sum_{i=1}^{m+1} p_i \cdot R_i - (1+\delta)^{-m-1} \leq \mathsf{val}(T') \leq (1+\delta) \cdot \sum_{i=1}^{m} p_i \cdot R_{i-1} \,. \tag{6.6}$$

Let C_i be a circuit that has $\mathbb{x}$ and the transcript tr hardwired such that $C_i(\alpha) = 1$ if and only if $R_i \leq \mathsf{val}(T'(\alpha)) \leq R_{i-1}$. Therefore, $\Pr_\alpha[C_i(\alpha)] = p_i$. We observe that $\mathsf{val}(T'(\alpha))$ can be computed in space $\mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ (see Section 2.1.4) and so the circuit C_i can be implemented in size $\mathrm{poly}\left(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r}, \log\left((1+\delta)^i\right)\right) = \mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ with PSPACE gates.

Then, by the pseudo-randomness of the PRG against $\{C_i\}_{i\in[m]}$ we get that for every $i \in [m]$:

$$|\Pr_s[C_i(\mathsf{G}(s)) = 1] - p_i| = |\Pr_s[C_i(\mathsf{G}(s)) = 1] - \Pr_\alpha[C_i(\alpha) = 1]| \leq \epsilon_{\mathsf{PRG}} \,.$$

Using these definitions we have that:

$$\sum_{i=1}^{m+1} \Pr_s[C_i(\mathsf{G}(s)) = 1] \cdot R_i \leq \mathbb{E}_s[\mathsf{val}(T'(\mathsf{G}(s)))] \leq \sum_{i=1}^{m+1} \Pr_s[C_i(\mathsf{G}(s)) = 1] \cdot R_{i-1} \,.$$

Since $\mathbb{E}_s[\mathsf{val}(T'(\mathsf{G}(s)))] = \mathsf{val}(T'_{\mathsf{G}})$, and by combining the pseudo-randomness of the PRG, we have

$$\sum_{i=1}^{m+1} (p_i - \epsilon_{\mathsf{PRG}}) \cdot R_i \leq \mathsf{val}(T'_{\mathsf{G}}) \leq \sum_{i=1}^{m+1} (p_i + \epsilon_{\mathsf{PRG}}) \cdot R_{i-1} \,.$$

By applying Equation 6.6, we get:

$$(1+\delta)^{-1} \cdot \mathsf{val}(T') - \epsilon_{\mathsf{PRG}} \cdot \sum_{i=1}^{m+1} R_i \leq \mathsf{val}(T'_{\mathsf{G}}) \leq (1+\delta)^{-m} + (1+\delta) \cdot \mathsf{val}(T') + \epsilon_{\mathsf{PRG}} \cdot \sum_{i=1}^{m+1} R_{i+1} \,.$$

Finally, we note that the sums $\sum_{i\in[m]} R_i = \sum_{i=1}^{m} (1+\delta)^{-i}$ and $\sum_{i\in[m]} R_{i-1} = \sum_{i=1}^{m} (1+\delta)^{-i+1}$ are geometric series bounded from above by $\frac{1+\delta-(1+\delta)^{-m}}{\delta} \leq \frac{2}{\delta}$. Therefore:

$$(1+\delta)^{-1} \cdot \mathsf{val}(T') - \frac{2\epsilon_{\mathsf{PRG}}}{\delta} \leq \mathsf{val}(T'_{\mathsf{G}}) \leq (1+\delta)^{-m} + (1+\delta) \cdot \mathsf{val}(T') + \frac{2\epsilon_{\mathsf{PRG}}}{\delta} \,.$$

We begin by refining the lower-bound:

$$\begin{aligned}(1+\delta)^{-1}\cdot\mathsf{val}(T')-\frac{2\epsilon_{\mathsf{PRG}}}{\delta} &= (1+\delta)^{-1}\cdot\left(\mathsf{val}(T')-\frac{2\epsilon_{\mathsf{PRG}}(1+\delta)}{\delta}\right)\\ &\geq (1+\delta)^{-1}\cdot\left(\mathsf{val}(T')-\frac{4\epsilon_{\mathsf{PRG}}}{\delta}\right)\\ &\geq (1+1/\mathsf{k})^{-1}\cdot\left(\mathsf{val}(T')-12\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}\right),\end{aligned}$$

where the final equality follows by recalling that $\delta=\frac{1}{3\mathsf{k}}$. We now refine the upper-bound, recalling that $(1+\delta)^m\cdot\delta\cdot\mathsf{val}(T')\geq 1$:

$$\begin{aligned}(1+\delta)^{-m}+(1+\delta)\cdot\mathsf{val}(T')+\frac{2\epsilon_{\mathsf{PRG}}}{\delta} &\leq \delta\cdot\mathsf{val}(T')+(1+\delta)\cdot\mathsf{val}(T')+\frac{2\epsilon_{\mathsf{PRG}}}{\delta}\\ &\leq (1+3\delta)\cdot\left(\mathsf{val}(T')+\frac{2\epsilon_{\mathsf{PRG}}}{\delta^2}\right)\\ &= (1+1/\mathsf{k})\cdot\left(\mathsf{val}(T')+18\cdot\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}^2\right).\end{aligned}$$

Putting all of this together, we conclude that

$$(1+1/\mathsf{k})^{-1}\cdot\left(\mathsf{val}(T')-12\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}\right)\leq\mathsf{val}(T'_{\mathsf{G}})\leq(1+1/\mathsf{k})\cdot\left(\mathsf{val}(T')+18\cdot\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}^2\right).$$

□

We now use Theorem 6.5.9 to show that the value of the interaction tree does not grow significantly over the entire interaction. Let T_i be the tree $T_{\mathbb{x}}$ after changing the nodes in layers $i,\ldots,\mathsf{k}$ to subsample from G. Notice that $T_0=T_{\mathbb{x},\mathsf{G}}$ and $T_{\mathsf{k}}=T_{\mathbb{x}}$. Then:

$$(1+1/\mathsf{k})^{-1}\cdot(\mathsf{val}(T_{i+1})-12\epsilon_{\mathsf{PRG}}\cdot\mathsf{k})\leq\mathsf{val}(T_i)\leq(1+1/\mathsf{k})\cdot\left(\mathsf{val}(T_{i+1})+18\cdot\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}^2\right).$$

This follows since, by Theorem 6.5.9, this is the variation in value of changing a node in layer i to sub-sample from G. Since all the nodes changed are in the same layer, the expected value of the tree also changes by the same value.

We now combine the bounds over all the layers to achieve the required bounds. We begin with the lower-bound:

$$\begin{aligned}\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) &\geq (1+1/\mathsf{k})^{-1}\cdot(\mathsf{val}(T_{\mathsf{k}-1})-12\epsilon_{\mathsf{PRG}}\cdot\mathsf{k})\\ &\geq\cdots\\ &\geq (1+1/\mathsf{k})^{-\mathsf{k}}\cdot(\mathsf{val}(T_0)-\mathsf{k}\cdot12\epsilon_{\mathsf{PRG}}\cdot\mathsf{k})\\ &\geq \mathsf{val}(T_{\mathbb{x}})/3-4\epsilon_{\mathsf{PRG}}\cdot\mathsf{k}^2.\end{aligned}$$

Similarly, we give the upper-bound:

$$\begin{aligned}\mathsf{val}(T_{\mathbb{x},\mathsf{G}}) &\leq (1+1/\mathsf{k}) \cdot \left(\mathsf{val}(T_{\mathsf{k}-1}) + 18 \cdot \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2\right) \\ &\leq \cdots \\ &\leq (1+1/\mathsf{k})^{\mathsf{k}} \cdot \left(\mathsf{val}(T_0) + \mathsf{k} \cdot 18 \cdot \epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2\right) \\ &\leq 3\mathsf{val}(T_{\mathbb{x}}) + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3 \,.\end{aligned}$$

Therefore, we conclude that:

$$\mathsf{val}(T_{\mathbb{x}})/3 - 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^2 \leq \mathsf{val}(T_{\mathbb{x},\mathsf{G}}) \leq 3\mathsf{val}(T_{\mathbb{x}}) + 54\epsilon_{\mathsf{PRG}} \cdot \mathsf{k}^3 \,.$$

□

6.6 Low-error IOPs to low-error PCPs

We show that IOPs with small soundness error imply PCPs with small soundness error. In particular (under complexity assumptions) in order to construct a sliding-scale PCP for NP, it suffices to construct an IOP for NP, with polynomial round complexity, with similar parameters.

We first show that this can be done using PRGs, meaning that, under complexity assumptions, IOPs can be transformed into PCPs.

Theorem 6.6.1. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ with (per round) interaction-randomness r. Suppose that there exists a PRG $\mathsf{G}\colon \{0,1\}^{\ell_{\mathsf{PRG}}} \to \{0,1\}^{\mathsf{r}}$ against circuits of size $\mathsf{s}_{\mathsf{PRG}} = \mathrm{poly}(|\mathbb{x}|, \mathsf{k}, \mathsf{l}, \mathsf{r})$ with* PSPACE *gates. Then $\mathcal{R}$ has a PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ with the following parameters:*

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

$+$

PRG G	
Seed length	ℓ_{PRG}
Error	ϵ_{PRG}
Security against size	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \mathsf{l}, \mathsf{r})$
Running time	$\mathsf{t}_{\mathsf{PRG}}$

$\longrightarrow$

PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$ where $\gamma := 1/3 \cdot (1-\alpha) \cdot (\mathsf{q}/(e \cdot \mathsf{k}))^{\mathsf{q}} - 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{q}^2$	
Completeness error	0
Soundness error	$O\left(\left(\beta + \epsilon_{\mathsf{PRG}} \cdot \mathsf{q}^3\right) \cdot \left(\frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{\gamma}\right)\right)$
Alphabet size	$\max\{\lambda, 2^{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}\}$
Length	$O\left(\mathsf{l} \cdot 2^{\mathsf{q} \cdot \ell_{\mathsf{PRG}}} \cdot \mathsf{q} \cdot \ell_{\mathsf{PRG}}/\gamma\right)$
Queries	$\mathsf{q} + 2$
Randomness	$O(\mathsf{q} \cdot \ell_{\mathsf{PRG}})$
Verifier running time	$\mathrm{poly}(\mathsf{vt}, \mathsf{q}, \mathsf{t}_{\mathsf{PRG}}, \log(\frac{\ell_{\mathsf{PRG}}}{\gamma}))$

We now show an analogue of the above theorem in which, rather than use a PRG, the verifier uses non-uniform advice.

Theorem 6.6.2. *Let $\mathcal{R}$ be a relation with a public-coin IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ and let $\varepsilon = \varepsilon(|\mathbb{x}|) \in (0,1]$ be a noticeable function. Then $\mathcal{R}$ has a PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ with the following parameters:*

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

$\longrightarrow$

PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$	
Completeness error	0
Soundness error	$O\left((\beta + \mathsf{q}\cdot\varepsilon)\cdot\left(\frac{\mathsf{q}\cdot\ell}{\gamma}\right)\right)$
Alphabet size	$\max\{\lambda, 2^{\mathsf{q}\cdot\ell_{\mathsf{PRG}}}\}$
Length	$O\left(\mathsf{l}\cdot 2^{\mathsf{q}\cdot\ell}\cdot\mathsf{q}\cdot\ell/\gamma\right)$
Queries	$\mathsf{q}+2$
Randomness	$O(\mathsf{q}\cdot\ell)$
Verifier running time	$\mathrm{poly}(\mathsf{vt},\mathsf{k},\mathsf{r},1/\varepsilon,\log(1/\gamma))$
Advice length	$O(\mathsf{q}\cdot\mathsf{k}\cdot\mathsf{r}\cdot\varepsilon^2)$

(Where $\gamma := (1-\alpha)\cdot(\mathsf{q}/(e\cdot\mathsf{k}))^{\mathsf{q}} - \mathsf{q}\cdot\varepsilon$ *and* $\ell := O\left(\log\left(\frac{\mathsf{q}\cdot\mathsf{r}\cdot\mathsf{k}\cdot|\mathbb{x}|}{\varepsilon}\right)\right) = O(\log(|\mathbb{x}|/\varepsilon))$*.)*

Corollary 6.6.3. *Let* $\mathcal{R}$ *be a relation with a public-coin IOP* $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ *with parameters as below. Then* $\mathcal{R}$ *has an* adaptive *PCP* $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ *with the following parameters:*

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ for $\mathcal{R}$	
Completeness error	0
Soundness error	$1/\lvert\mathbb{x}\rvert$
Rounds	$\mathrm{polylog}(\lvert\mathbb{x}\rvert)$
Alphabet size	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Proof length (per round)	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Queries	$O(1)$
Randomness (per round)	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

$\longrightarrow$

PCP with non-uniform advice $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$	
Completeness error	0
Soundness error	$1/\lvert\mathbb{x}\rvert$
Alphabet size	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Length	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Queries	$O(1)$
Randomness	$O(\log\lvert\mathbb{x}\rvert)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$
Advice length	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

Moreover, the non-uniform advice can be removed under the assumption that there exists a function in E *with circuit complexity* $2^{\Omega(n)}$ *for circuits with* PSPACE *gates.*

Proof. We first run amplify the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ twice so that the soundness error is $1/|\mathbb{x}|^2$, and all other parameters are related by at most a factor of 2. We then apply Theorem 6.6.2 with $\varepsilon = 1/|\mathbb{x}|^2$. Notice that here $\gamma = 1/\,\mathrm{polylog}(|\mathbb{x}|)$ and $\ell := O(\log|\mathbb{x}|)$. Therefore, the length of the PCP is equal to $\mathrm{poly}(|\mathbb{x}|\cdot 2^{O(\log|\mathbb{x}|)}\cdot \mathrm{polylog}(|\mathbb{x}|)) = \mathrm{poly}(|\mathbb{x}|)$, and it has soundness error $(1/|\mathbb{x}|^2 + 1/\mathrm{poly}(|\mathbb{x}|))\cdot\mathrm{polylog}(|\mathbb{x}|) \le 1/|\mathbb{x}|$.

If we assume that there exists a function in E with circuit complexity $2^{\Omega(n)}$ for circuits with PSPACE gates, then the non-uniform advice can be removed as follows: By Theorem 2.6.3, there exists a PRG $\mathsf{G}\colon \{0,1\}^{O(\log|\mathbb{x}|)} \to \{0,1\}^{\mathsf{r}}$ against $\mathrm{poly}(|\mathbb{x}|)$-sized circuits with PSPACE-gates and error $1/\mathrm{poly}(|\mathbb{x}|) \leq 1/|\mathbb{x}|^2$ (where $\mathsf{r} = \mathrm{poly}(|\mathbb{x}|)$ is the maximum number of random bits sent by the verifier of the amplified IOP in a round). We can then apply Theorem 6.6.1 in place of Theorem 6.6.2. □

Proof of Theorems 6.6.1 and 6.6.2. We first prove Theorem 6.6.1. We apply a sequence of transformations:

1. reduce the number of rounds of the IOP via Theorem 6.3.3 with $\ell := \mathsf{q}$;
2. derandomize the IOP verifier via Theorem 6.5.6;
3. transform the IOP to have perfect completeness via Theorem 6.4.3; and
4. unroll the IOP into a PCP via Theorem 6.3.6.

This sequence of transformations and their effects on the parameters of the resulting proof systems is described in the list of tables below. The proof of Theorem 6.6.1 uses an identical series of transformations, except that Theorem 6.5.6 is replaced with Theorem 6.5.1 (and as a result, the parameters change slightly).

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ *for* $\mathcal{R}$	
Completeness error	α
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	r
Verifier running time	vt

↓ *(Item 1)*

Round-reduced IOP for $\mathcal{R}$ *via Theorem 6.3.3 with* $\ell := \mathsf{q}$	
Completeness error	$1-(1-\alpha)\cdot(\mathsf{q}/(e\cdot\mathsf{k}))^{\mathsf{q}}$
Soundness error	β
Rounds	q
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	$\mathsf{k}\cdot\mathsf{r}+\mathsf{q}\cdot\log\mathsf{k} < (\mathsf{q}+1)\cdot\mathsf{k}\cdot\mathsf{r}$
Verifier running time	$\mathrm{poly}(\mathsf{vt})$

↓ *(Item 2)*

Derandomized IOP for $\mathcal{R}$ via Theorem 6.5.6 with PRG G with $\gamma := 1/3 \cdot (1-\alpha) \cdot (\mathsf{q}/(e \cdot \mathsf{k}))^{\mathsf{q}} - 4\epsilon_{\mathsf{PRG}} \cdot \mathsf{q}^2$	
Completeness error	$1-\gamma$
Soundness error	$O(\beta + \varepsilon \cdot \mathsf{q}^3)$
Rounds	q
Alphabet size	λ
Proof length (per round)	l
Queries	q
Randomness (per round)	ℓ_{PRG}
Verifier running time	$\mathrm{poly}(\mathsf{vt}, \mathsf{q}, \mathsf{t}_{\mathsf{PRG}})$

↓ *(Item 3)*

Perfectly complete IOP for $\mathcal{R}$ via Theorem 6.4.3	
Completeness error	0
Soundness error	$O\left((\beta + \epsilon_{\mathsf{PRG}} \cdot \mathsf{q}^3) \cdot \left(\frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)\right)$
Rounds	$\mathsf{q}+1$
Alphabet size	$\max\{\lambda, 2^{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}\}$
Proof length (per round)	$O\left(\mathsf{l} \cdot \frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)$
Queries	$\mathsf{q}+2$
Randomness (per round)	ℓ_{PRG}
Verifier running time	$\mathrm{poly}\left(\mathsf{vt}, \mathsf{q}, \mathsf{t}_{\mathsf{PRG}}, \log\left(\frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)\right)$

↓ *(Item 4)*

PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$ via Theorem 6.3.6	
Completeness error	0
Soundness error	$O\left((\beta + \epsilon_{\mathsf{PRG}} \cdot \mathsf{q}^3) \cdot \left(\frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)\right)$
Alphabet size	$\max\{\lambda, 2^{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}\}$
Length	$O\left(\mathsf{l} \cdot 2^{\mathsf{q} \cdot \ell_{\mathsf{PRG}}} \cdot \frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)$
Queries	$\mathsf{q}+2$
Randomness	$\mathsf{q} \cdot \ell_{\mathsf{PRG}}$
Verifier running time	$\mathrm{poly}\left(\mathsf{vt}, \mathsf{q}, \mathsf{t}_{\mathsf{PRG}}, \log\left(\frac{\mathsf{q} \cdot \ell_{\mathsf{PRG}}}{-\log(1-\gamma)}\right)\right)$

Finally, notice that $-\log(1-\gamma) \geq \gamma$ for $0 < \gamma < 1$, and so $\frac{1}{-\log(1-\gamma)} \leq \frac{1}{\gamma}$. □

6.7 Limitations of short IOPs

We show that, under the randomized exponential time hypothesis, there are no short IOPs for 3SAT with polylog(n) rounds and small soundness error.

Theorem 6.7.1. *Assume that* 3SAT $\notin$ BPTIME$[2^{c\cdot n}]$ *for a constant* $c > 0$ *and let* $(\mathbf{P}, \mathbf{V})$ *be a public-coin IOP for* 3SAT *with the following parameters:*

Public-coin IOP $(\mathbf{P}, \mathbf{V})$ for n-variate 3SAT	
Completeness error	0
Soundness error	β
Rounds	polylog(n)
Alphabet size	λ
Proof length (total)	$\mathsf{l}_{\mathrm{tot}}$
Queries	q
Randomness (per round)	poly(n)
Verifier running time	poly(n)

If $\mathsf{l}_{\mathrm{tot}} \geq n$ *and* $\left(\frac{\mathsf{l}_{\mathrm{tot}} \cdot \log \lambda}{n}\right)^{\mathsf{q}} \leq n^{\mathrm{polylog}(n)}$ *then* $\beta > \frac{1}{2} \cdot \left(\frac{2 \cdot e \cdot \mathsf{l}_{\mathrm{tot}} \cdot \log \lambda}{c \cdot n}\right)^{-\mathsf{q}}$.

Proof of Theorem 6.7.1. We reduce the length of the IOP $(\mathbf{P}, \mathbf{V})$ using Theorem 6.3.1. Let $m = \mathsf{l}_{\mathrm{tot}}/n$. The parameters of the resulting IOP $(\mathbf{P}', \mathbf{V}')$ are described in the following tables:

IOP $(\mathbf{P}, \mathbf{V})$ *for n-variate* 3SAT	
Completeness error	0
Soundness error	β
Rounds	polylog(n)
Alphabet size	λ
Proof length (total)	$n \cdot m$
Queries	q
Randomness (per round)	poly(n)
Verifier running time	poly(n)

↓

Length-reduced IOP $(\mathbf{P}', \mathbf{V}')$ *for n-variate* 3SAT *via Theorem 6.3.1 with* $\ell := e \cdot n \cdot m \cdot (2\beta)^{1/\mathsf{q}}$	
Completeness error	$\alpha' := 1 - 2\beta$
Soundness error	$\beta' := \beta$
Rounds	$\mathsf{k}' :=$ polylog(n)
Alphabet size	λ
Proof length (total)	$\mathsf{l}'_{\mathrm{tot}} := e \cdot n \cdot m \cdot (2\beta)^{1/\mathsf{q}}$
Queries	q
Randomness (per round)	poly(n)
Verifier running time	poly(n)

We now apply Theorem 2.1.3 with the IOP $(\mathbf{P}', \mathbf{V}')$ (viewed as an IP) to construct an algorithm for 3SAT. When considering the IOP with a binary alphabet (by writing each symbol in its binary representation), Theorem 2.1.3 says that for

$$\begin{aligned} d &= \mathsf{l}'_{\mathrm{tot}} \cdot \log \lambda + \mathsf{k}' \log \mathsf{k}' - \mathsf{k}' \cdot \log\left(1 - \alpha' - \beta'\right) + O(\log n) \\ &= e \cdot n \cdot m \cdot \log \lambda \cdot (2\beta)^{1/\mathsf{q}} + O(\mathrm{polylog}(n) \cdot \log 1/\beta) + o(n)\,, \end{aligned}$$

there exists an algorithm that decides 3SAT in probabilistic time $2^{O(d)}$. Suppose towards contradiction (of RETH) that $\beta \leq \frac{1}{2} \cdot (2 \cdot e \cdot m \cdot \log \lambda / c)^{-\mathsf{q}}$. Notice that $\frac{1}{2} \cdot (2 \cdot e \cdot m \cdot \log \lambda / c)^{-\mathsf{q}} \geq n^{-\mathrm{polylog}(n)}$, thus we can assume (without loss of generality) that $\beta \geq n^{-\mathrm{polylog}(n)}$, and, as a result, that $\mathrm{polylog}(n) \cdot \log 1/\beta = o(n)$. Then we have that:

$$\begin{aligned} d &= e \cdot n \cdot m \cdot \log \lambda \cdot (2\beta)^{1/\mathsf{q}} + o(n) \\ &= c \cdot n/2 + o(n) \\ &< c \cdot n\,. \end{aligned}$$

As a result, we have an algorithm that decides 3SAT in time $2^{c \cdot n}$ in contradiction to the (RETH) assumption that 3SAT $\notin$ BPTIME$[2^{c \cdot n}]$. □

6.8 Limitations of high-round low-query IOPs

We show that if a relation cannot be decided by IPs with small round complexity, then any round-query IOP with small round-query complexity for the relation cannot have small soundness error.

Theorem 6.8.1. *Let* k *and* ℓ *be parameters and* $\mathcal{R}$ *be a relation where* $\mathcal{R} \in \mathrm{AM}[\mathsf{k}] \backslash \mathrm{AM}[\ell]$. *Suppose that* $\mathcal{R}$ *has a* k*-round public-coin round-query IOP* $(\mathbf{P}, \mathbf{V})$ *with completeness error* α*, soundness error* β*, and round-query complexity* $\mathsf{q}_{\mathsf{round}} \leq \ell$.

Then $\beta \geq (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k}))^{\mathsf{q}_{\mathsf{round}}} - |\mathbb{x}|^{-c}$ *for every constant* $c > 0$.

Proof of Theorem 6.8.1. Apply Theorem 6.3.3 to the k-round IOP $(\mathbf{P}, \mathbf{V})$ with parameter ℓ. This results in a ℓ-round IOP $(\mathbf{P}', \mathbf{V}')$ with completeness error $\alpha' := 1 - (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k}))^{\mathsf{q}_{\mathsf{round}}}$ and soundness error β. Suppose towards contradiction that $\beta < (1-\alpha) \cdot (\ell/(e \cdot \mathsf{k}))^{\mathsf{q}_{\mathsf{round}}} - |\mathbb{x}|^{-c}$ for some $c \in \mathbb{N}$. Then the gap between completeness and soundness error of $(\mathbf{P}', \mathbf{V}')$ is $1 - \alpha' - \beta > |\mathbb{x}|^{-c}$.

Since the gap between completeness and soundness error of $(\mathbf{P}', \mathbf{V}')$ is polynomial $(\mathbf{P}', \mathbf{V}')$ can be transformed into a ℓ-round public-coin IP $(\mathbf{P}'', \mathbf{V}'')$ for $\mathcal{R}$ with completeness error $1/3$ and soundness error $1/3$. This is done by using the standard technique of taking $\mathrm{poly}(|\mathbb{x}|)$ parallel repetitions, computing the fraction of accepting transcripts, and accepting if the number of accepting transcripts is beyond some threshold that depends on α' and $|\mathbb{x}|^{-c}$. The IP $(\mathbf{P}'', \mathbf{V}'')$, then, contradicts the assumption that $\mathcal{R} \notin \mathrm{AM}[\ell]$.

□

6.9 Limitations of binary-alphabet constant-query IOPs

We prove lower bounds on the soundness error of 2-query and 3-query IOPs for 3SAT over the binary alphabet.

Theorem 6.9.1. *Assume that* RETH *holds and suppose that there exists a non-adaptive public-coin IOP* $(\mathbf{P}, \mathbf{V})$ *for* 3SAT *with the following parameters:*

Non-adaptive public-coin IOP $(\mathbf{P}, \mathbf{V})$ for n-variate 3SAT	
Completeness error	0
Soundness error	β
Rounds	k
Alphabet size	2
Proof length (per round)	$2^{o(n)}$
Queries	q
Randomness (per round)	r
Verifier running time	$2^{o(n)}$

Then:

- *If* $\mathsf{q} = 2$ *then* $\beta > 1 - \varepsilon$ *for every* ε *with* $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$.
- *If* $\mathsf{q} = 3$ *then* $\beta > 5/8 - \varepsilon$ *for every* ε *with* $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$.

The theorem follows from a generic lemma that we prove in two steps. In Section 6.9.1 we formalize the folklore idea that, in certain parameter regimes, it is unlikely that there simultaneously exist CSP solvers and PCPs. In Section 6.9.2 we build on this and prove an analogous lemma for IOPs.

When compared to the following analogous (folklore) theorem for binary-alphabet PCPs, Theorem 6.9.1 shows that interaction does not grant additional power when restricted to 2 or 3 queries. The soundness beyond which RETH is contradicted is nearly identical to that of PCPs (in order to contradict ETH). Moreover, the fewer rounds the IOP has, the smaller this gap.

Theorem 6.9.2 (folklore)**.** *Assume that the* ETH *conjecture holds and suppose that there exists a non-adaptive PCP* $(\mathbf{P}, \mathbf{V})$ *for* 3SAT *with the following parameters:*

Non-adaptive PCP $(\mathbf{P}, \mathbf{V})$ for n-variate 3SAT	
Completeness error	0
Soundness error	β
Alphabet size	2
Length	$2^{o(n)}$
Queries	q
Randomness	$o(n)$
Verifier running time	$2^{o(n)}$

Then:

- *If* $\mathsf{q} = 2$ *then* $\beta = 1$.
- *If* $\mathsf{q} = 3$ *then* $\beta \geq 5/8$.

The proof of Theorem 6.9.1 uses the fact that there exist efficient algorithms for deciding gap problems for CSPs with arities 2 and 3. This proof is generic in the sense that, given an efficient algorithm for deciding gap problems for CSPs with arity q, our proof gives a lower-bound on the soundness error of any IOP for 3SAT with query complexity q.

6.9.1 Algorithms for 3SAT from CSP-solvers and PCPs

We prove a generic lemma that states that, for certain parameters, the simultaneous existence of a PCP for 3SAT and an algorithm for solving CSPs implies a fast algorithm for 3SAT.

Lemma 6.9.3. *Suppose that the following exist:*

- *A non-adaptive PCP* $(\mathbf{P}, \mathbf{V})$ *for* n*-variable* 3SAT *with perfect completeness, soundness error* β*, alphabet* Σ*, proof length* $\mathsf{l} = 2^{o(n)}$ *, query complexity* q*, randomness complexity* $\mathsf{r} = o(n)$*, and verifier running time* $\mathsf{vt} = 2^{o(n)}$ *given* $\ell(n)$ *bits of non-uniform advice.*
- *A deterministic algorithm* $\mathbf{A}$ *that decides in time* $\mathrm{poly}(|\psi|)$ *whether an input* (Σ, q)*-CSP instance* ψ *has value* 1 *or value at most* β*.*

Then there exists an algorithm $\mathbf{A}'$ *that decides whether a* 3SAT *formula* ϕ *over* n *variables is satisfiable in time* $2^{o(n)}$ *given* $\ell(n)$ *bits of non-uniform advice. (Moreover, any advice string that is good for the PCP is also good for the resulting algorithm.)*

Proof. The algorithm $\mathbf{A}'$ receives as input a 3SAT formula ϕ on n variables and a non-uniform advice string $z \in \{0,1\}^{\ell}$, and works as follows.

1. Use Theorem 2.5.3 to transform the PCP $(\mathbf{P}, \mathbf{V})$ on input the 3SAT formula ϕ and advice string z into a (Σ, q)-CSP instance ψ.
2. Run $\mathbf{A}(\psi)$ and output that ϕ is satisfiable if and only if $\mathbf{A}$ outputs that the value of ψ is 1.

We analyze the correctness and running time of the algorithm $\mathbf{A}'$.

- *Correctness.* By Theorem 2.5.3, ψ is a (Σ, q)-CSP with size $\mathrm{poly}(2^{\mathsf{r}}, \mathsf{vt}) = 2^{o(n)}$ where: (i) if $\phi \in$ 3SAT then ψ has value 1; (ii) if $\phi \notin$ 3SAT then ψ has value at most β. By the correctness of the algorithm $\mathbf{A}$ we conclude that the algorithm $\mathbf{A}'$ correctly decides whether ϕ is satisfiable.
- *Running time.* The algorithm $\mathbf{A}'$ runs in time $2^{o(n)}$, because its first step runs in time $\mathrm{poly}(2^{\mathsf{r}}, \mathsf{vt}) = 2^{o(n)}$ and its second step runs in time $\mathrm{poly}(|\psi|) = 2^{o(n)}$.

□

6.9.2 Algorithms for 3SAT from CSP-solvers and IOPs

We prove a generic lemma that states that, for certain parameters, the simultaneous existence of an IOP for 3SAT and an algorithm for solving CSPs implies a fast *probabilistic* algorithm for 3SAT.

Lemma 6.9.4. *Let* k, r *and* ε *be parameters with* $\mathsf{k} \cdot \log(\mathsf{r}/\varepsilon) = o(n)$. *Suppose that the following exist:*

- *A non-adaptive public-coin IOP* $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ *for* n*-variable* 3SAT *with perfect completeness, soundness error* β, *round complexity* k, *alphabet* Σ, *per-round proof length* $\mathsf{l} = 2^{o(n)}$, *query complexity* q, *per-round randomness complexity* r, *and verifier running time* $\mathsf{vt} = 2^{o(n)}$.
- *A deterministic algorithm* $\mathbf{A}$ *that decides in time* $\mathrm{poly}(|\psi|)$ *whether an input* (Σ, q)*-CSP instance* ψ *has value* 1 *or value at most* $\beta + \varepsilon$.

Then there exists a probabilistic algorithm $\mathbf{A}'$ *that decides whether a* 3SAT *formula* ϕ *over* n *variables is satisfiable in time* $2^{o(n)}$ *with probability at least* $2/3$.

Proof. First, we transform the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ into a PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ for 3SAT as follows: (i) use Theorem 6.5.1 to derandomize the IOP using non-uniform advice; and (ii) use Theorem 6.3.6 to unroll the IOP into a PCP. Moreover, in Theorem 6.5.1 we may sample a random advice string rather than using non-uniformity: a random advice string is good with probability at least $1 - 2^{|\phi|}$. This same advice string can be used in the PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ derived by applying Theorem 6.3.6.

Next, we apply Theorem 6.9.3 with the PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ and the algorithm $\mathbf{A}$ to derive an algorithm $\mathbf{A}^*$ for 3SAT for whom the same advice strings as the PCP are good. Notice that Theorem 6.9.3 requires a PCP of length $2^{o(n)}$and that the PCP $(\mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}})$ resulting from the series of transformations has proof length $2^{O(\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon)) + o(n)}$. This length is sub-exponential in n since we require $\mathsf{k} \cdot \log(\mathsf{r} \cdot n/\varepsilon) = o(n)$.

Finally, we have that the algorithm $\mathbf{A}'$ that, on input a 3SAT formula ϕ first samples a random advice string z and then outputs the same as $\mathbf{A}^*(\phi, z)$ succeeds in deciding 3SAT with probability at least $1 - 2^{-|\phi|} \geq 2/3$.

In order to keep track of parameters, we describe the transformation as a series of tables :

IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ *for* $\mathcal{R}$	
Completeness error	0
Soundness error	β
Rounds	k
Alphabet size	λ
Proof length (per round)	$2^{o(n)}$
Queries	q
Randomness (per round)	r
Verifier running time	$2^{o(n)}$

↓

Derandomized IOP for $\mathcal{R}$ via Theorem 6.5.1 with error ε/k	
Completeness error	0
Soundness error	$\beta+\varepsilon$
Rounds	k
Alphabet size	λ
Proof length (per round)	$2^{o(n)}$
Queries	q
Randomness (per round)	$O(\log(\mathsf{r}\cdot n/\varepsilon))$
Verifier running time	$2^{o(n)}+\mathrm{poly}(n,\mathsf{r},1/\varepsilon)$
Advice length (chosen at random)	$\mathrm{poly}(n,\mathsf{r},1/\varepsilon)$

↓

PCP $(\mathbf{P}_{\mathsf{PCP}},\mathbf{V}_{\mathsf{PCP}})$ for $\mathcal{R}$ via Theorem 6.3.6	
Completeness error	0
Soundness error	$\beta+\varepsilon$
Alphabet size	λ
Length	$2^{O(\mathsf{k}\cdot\log(\mathsf{r}\cdot n/\varepsilon))+o(n)}$
Queries	q
Randomness	$O(\log(\mathsf{r}\cdot n/\varepsilon))$
Verifier running time	$2^{o(n)}+\mathrm{poly}(n,\mathsf{r},1/\varepsilon)$
Advice length (chosen at random)	$\mathrm{poly}(\mathsf{r}/\varepsilon)$

□

6.9.3 Proof of Theorems 6.9.1 and 6.9.2

We first prove Theorem 6.9.1. Consider an IOP with parameters as in the statement of Theorem 6.9.1. Suppose that there exist γ and ε with $\mathsf{k}\cdot\log(\mathsf{r}/\varepsilon)=o(n)$ such that: (i) the IOP soundness error β is at most $\gamma-\varepsilon$; and (ii) there exists an algorithm $\mathbf{A}$ that decides in time $\mathrm{poly}(|\psi|)$ whether an input $(\{0,1\},\mathsf{q})$-CSP instance ψ has value 1 or value at most γ. Then, by Theorem 6.9.4, there exists a probabilistic algorithm that decides 3SAT in time $2^{o(n)}$, in contradiction to RETH. Therefore, assuming RETH, the above items cannot both be true.

We conclude the proof of Theorem 6.9.1 by relying on known algorithms for solving CSPs with appropriate arities q and decision bounds γ, implying that (assuming RETH) it must be that $\beta>\gamma-\varepsilon$ for every ε such that $\mathsf{k}\cdot\log(\mathsf{r}/\varepsilon)=o(n)$.

- For $\mathsf{q}=2$, we rely on Schaefer's dichotomy theorem (Theorem 2.5.4), which says that the satisfiability of a $(\{0,1\},2)$-CSP can be decided in polynomial time. In this case $\gamma=1$.

- For $\mathsf{q}=3$, we rely on Zwick's algorithm (Theorem 2.5.5), which decides whether a $(\{0,1\},3)$-CSP has value 1 or value smaller than $5/8$ in polynomial time. In this case $\gamma=5/8$.

Finally, the proof of Theorem 6.9.2 is identical to the one of Theorem 6.9.1, except that Theorem 6.9.3 is used in place of Theorem 6.9.4. In more detail, under ETH, for every γ the following cannot be true simultaneously; (i) the PCP soundness error β is at most γ; and (ii) there exists an algorithm $\mathbf{A}$ that decides in time $\text{poly}(|\psi|)$ whether an input $(2, q)$-CSP instance ψ has value 1 or value at most γ. As in the proof of Theorem 6.9.1, we rely on Schaefer's dichotomy theorem and Zwick's algorithm to obtain the claimed lower bounds on β.

References

[AASY16] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. "Incompressible Functions, Relative-Error Extractors, and the Power of Nondeterministic Reductions". In: *Computational Complexity* 25.2 (2016), pp. 349–418.

[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.

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

[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.

[ACY23] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "IOPs with Inverse Polynomial Soundness Error". In: *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*. IEEE, 2023, pp. 752–761.

[AG21] Benny Applebaum and Eyal Golombek. "On the Randomness Complexity of Interactive Proofs and Statistical Zero-Knowledge Proofs". In: *Proceedings of the 2nd Conference on Information-Theoretic Cryptography*. ITC '21. 2021, 4:1–4:23.

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. "Pseudorandomness When the Odds are Against You". In: *Proceedings of the 31st Annual Conference on Computational Complexity*. CCC '16. 2016, 9:1–9:35.

[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.

[BGG90] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. "Randomness in Interactive Proofs". In: *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. FOCS '90. 1990, pp. 563–572.

[BGHSV05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Short PCPs Verifiable in Polylogarithmic Time". In: *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*. CCC '05. 2005, pp. 120–134.

[BGHSV06a] 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.

[BGHSV06b] 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.

[BGKTRTZ23] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. "Fiat-Shamir Security of FRI and Related SNARKs". In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14439. Lecture Notes in Computer Science. Springer, 2023, pp. 3–40.

[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.

[BGTZ23] Alexander R. Block, Albert Garreta, Pratyush Ranjan Tiwari, and Michal Zajac. "On Soundness Notions for Interactive Oracle Proofs". In: *IACR Cryptol. ePrint Arch.* (2023), p. 1256.

[BM88] László Babai and Shlomo Moran. "Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes". In: *Journal of Computer and System Sciences* 36.2 (1988), pp. 254–276.

[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.

[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.

[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.

[CCHLRR18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. *Fiat–Shamir From Simpler Assumptions*. Cryptology ePrint Archive, Report 2018/1004. 2018.

[CCHLRRW19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Ed. by Moses Charikar and Edith Cohen. ACM, 2019, pp. 1082–1090.

[CFLS95] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter W. Shor. "Probabilistically Checkable Debate Systems and Nonapproximability of PSPACE-Hard Functions". In: *Chicago Journal of Theoretical Computer Science* 1995 (1995).

[CFLS97] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter W. Shor. "Random Debaters and the Hardness of Approximating Stochastic Functions". In: *SIAM Journal on Computing* 26.2 (1997), pp. 369–400.

[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.

[CIKP08] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. "The complexity of Unique k-SAT: An Isolation Lemma for k-CNFs". In: *J. Comput. Syst. Sci.* 74.3 (2008), pp. 386–393.

[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.

[CY20] Alessandro Chiesa and Eylon Yogev. "Barriers for Succinct Arguments in the Random Oracle Model". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 47–76.

[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.

[DH13] Irit Dinur and Prahladh Harsha. "Composition of Low-Error 2-Query PCPs Using Decodable PCPs". In: *SIAM Journal on Computing* 42.6 (2013). Preliminary version appeared in Property Testing '10., pp. 2452–2486.

[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.

[DL78] Richard A. DeMillo and Richard J. Lipton. "A Probabilistic Remark on Algebraic Program Testing". In: *Information Processing Letters* 7.4 (1978), pp. 193–195.

[DR04] Irit Dinur and Omer Reingold. "Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem". In: *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '04. 2004, pp. 155–164.

[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.

[Dru11a] Andrew Drucker. "A PCP Characterization of AM". In: *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*. ICALP '11. 2011, pp. 581–592.

[Dru11b] Andrew Drucker. "Efficient Probabilistically Checkable Debates". In: *Proceedings of the 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization*. RANDOM '11. 2011, pp. 519–529.

[FGLSS91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. "Approximating clique is almost NP-complete (preliminary version)". In: *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*. SFCS '91. 1991, pp. 2–12.

[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.

[FGMSZ89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. "On Completeness and Soundness in Interactive Proof Systems". In: *Advances in Computing Research* 5 (1989), pp. 429–442.

[FS11] Lance Fortnow and Rahul Santhanam. "Infeasibility of instance compression and succinct PCPs for NP". In: *J. Comput. Syst. Sci.* 77.1 (2011), pp. 91–106. DOI: 10.1016/J.JCSS.2010.06.007. URL: https://doi.org/10.1016/j.jcss.2010.06.007.

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Proceedings of the 6th Annual International Cryptology Conference*. CRYPTO '86. 1986, pp. 186–194.

[GH98] Oded Goldreich and Johan Håstad. "On the complexity of interactive proofs with bounded communication". In: *Information Processing Letters* 67.4 (1998), pp. 205–214.

[GI05] Venkatesan Guruswami and Piotr Indyk. "Linear-time encodable/decodable codes with near-optimal rate". In: *IEEE Transactions on Information Theory* 51.10 (2005). Preliminary version appeared in STOC '03., pp. 3393–3400.

[GKKRRS19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. *Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems*. IACR Cryptology ePrint Archive, Report 2019/458. 2019.

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

[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.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems". In: *Journal of the ACM* 38.3 (1991). Preliminary version appeared in FOCS '86., pp. 691–729.

[GRS23] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. 2023. URL: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. "Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes". In: *Journal of the ACM* 56.4 (2009), 20:1–20:34.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "On interactive proofs with a laconic prover". In: *Computational Complexity* 11.1/2 (2002), pp. 1–53.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol98] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Vol. 17. Algorithms and Combinatorics. Springer, 1998.

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. "Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge)". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 750–760.

[HRT07] Ishay Haviv, Oded Regev, and Amnon Ta-Shma. "On the Hardness of Satisfiability with Bounded Occurrences in the Polynomial-Time Hierarchy". In: *Theory of Computing* 3.1 (2007), pp. 45–60.

[Has05] Gustav Hast. "Beating a random assignment: Approximating constraint satisfaction problems". PhD thesis. KTH, 2005.

[Hås14] Johan Håstad. "On the NP-hardness of Max-Not-2". In: *SIAM Journal on Computing* 43.1 (2014), pp. 179–193.

[Her] *Hermez*. https://hermez.io.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which Problems Have Strongly Exponential Complexity?" In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530.

[IW14] Yuval Ishai and Mor Weiss. "Probabilistically Checkable Proofs of Proximity with Zero-Knowledge". In: *Proceedings of the 11th Theory of Cryptography Conference*. TCC '14. 2014, pp. 121–145.

[IW97] Russell Impagliazzo and Avi Wigderson. "P=BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. STOC '97. 1997, pp. 220–229.

[Joh62] S. Johnson. "A new upper bound for error-correcting codes". In: *IRE Transactions on Information Theory* 8.3 (1962), pp. 203–207.

[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.

[Kil92] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. STOC '92. 1992, pp. 723–732.

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

[MNT22] Pasin Manurangsi, Preetum Nakkiran, and Luca Trevisan. "Near-Optimal NP-Hardness of Approximating MAX k-CSPR". In: *Theory of Computing* 18.3 (2022), pp. 1–29.

[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.

[McD89] Colin McDiarmid. "On the method of bounded differences". In: *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*. Ed. by J.Editor Siemons. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989, 148–188.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS '94., pp. 1253–1298.

[Mid] *Miden*. https://github.com/0xPolygonMiden.

[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.

[Mul54] David E. Muller. "Application of Boolean algebra to switching circuit design and to error detection". In: *Trans. I R E Prof. Group Electron. Comput.* 3.3 (1954), pp. 6–12.

[NR22] Shafik Nassar and Ron D. Rothblum. "Succinct Interactive Oracle Proofs: Applications and Limitations". In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO '22. 2022.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs Randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[Nep] *Netpture*. https://neptune.cash/.

[Ola] *Ola*. https://ola.finance. Accessed: insert date here.

[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.

[Pol] *Polygon*. https://polygon.technology.

[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.

[RS60] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304.

[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.

[RV09] Guy N. Rothblum and Salil Vadhan. "Are PCPs Inherent in Efficient Arguments?" In: *Proceedings of the 24th IEEE Annual Conference on Computational Complexity*. CCC '09. 2009, pp. 81–92.

[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.

[Ree54] Irving S. Reed. "A class of multiple-error-correcting codes and the decoding scheme". In: *Transactions of the IRE Professional Group on Information Theory* 4 (1954), pp. 38–49.

[Ris] *Risc0*. https://risc0.com.

[San] *Sandstorm*. https://github.com/andrewmilson/sandstorm.

[Sch78] Thomas J. Schaefer. "The Complexity of Satisfiability Problems". In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*. STOC '78. 1978, pp. 216–226.

[Sch80] Jacob T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *Journal of the ACM* 27.4 (1980), pp. 701–717.

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

[Staa] *StarkEx*. https://starkware.co/starkex/.

[Stab] *StarkNet*. https://www.starknet.io/.

[Sta21] StarkWare. *ethSTARK Documentation*. Cryptology ePrint Archive, Paper 2021/582. https://eprint.iacr.org/2021/582. 2021. URL: https://eprint.iacr.org/2021/582.

[Wil05] Ryan Williams. "A new algorithm for optimal 2-constraint satisfaction and its implications". In: *Theor. Comput. Sci.* 348.2-3 (2005), pp. 357–365.

[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.

[Zip79] Richard Zippel. "Probabilistic algorithms for sparse polynomials". In: *Proceedings of the 1979 International Symposium on Symbolic and Algebraic Computation*. EUROSAM '79. 1979, pp. 216–226.

[Zks] *zkSync*. https://zksync.io.

[Zwi98] Uri Zwick. "Approximation Algorithms for Constraint Satisfaction Problems Involving at Most Three Variables per Constraint". In: *Proceedings of the 9th Annual Symposium on Discrete Algorithms*. SODA '98. 1998, pp. 201–210.

[ark] arkworks. *An ecosystem for developing and programming with zkSNARKs*. arkworks.rs.