STIR: Reed–Solomon Proximity Testing with Fewer Queries

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

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

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

March 3, 2024

Abstract

We present STIR (Shift To Improve Rate), an interactive oracle proof of proximity (IOPP) for Reed–Solomon codes that achieves the best known query complexity of any concretely efficient IOPP for this problem. For λ bits of security, STIR has query complexity $O(\log d + \lambda \cdot \log\log d)$, while FRI, a popular protocol, has query complexity $O(\lambda \cdot \log d)$ (including variants of FRI based on conjectured security assumptions). STIR relies on a new technique for recursively improving the rate of the tested Reed–Solomon code.

We provide an implementation of STIR compiled to a SNARK. Compared to a highly-optimized implementation of FRI, STIR achieves an improvement in argument size that ranges from 1.25× to 2.46× depending on the chosen parameters, with similar prover and verifier running times. For example, in order to achieve 128 bits of security for degree 2^{26} and rate 1/4, STIR has argument size 114 KiB, compared to 211 KiB for FRI.

Keywords: interactive oracle proofs; Reed–Solomon proximity testing

Contents

1 Introduction

Reed–Solomon (RS) codes [RS60] are a fundamental object of study in algebraic coding theory and theoretical computer science; in particular, they often play a notable role in the design of proof systems. For a finite field $\mathbb{F}$, evaluation domain $\mathcal{L} \subseteq \mathbb{F}$, and degree bound d, the Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ consists of all functions $f\colon \mathcal{L} \to \mathbb{F}$ obtained by evaluating on $\mathcal{L}$ a polynomial of degree (strictly) smaller than d over $\mathbb{F}$. The rate $\rho := d/|\mathcal{L}|$ of the code represents (the inverse of) the relative amount of redundancy of the code.

The RS proximity testing problem considers the setting where a verifier has query access to a function $f\colon \mathcal{L} \to \mathbb{F}$ and the goal is to distinguish, by querying f at few locations, whether f 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 [Din07; BS08; Mie09; BGHSV06] 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].

[1] More precisely, the BCS transformation requires a notion called state-restoration soundness, which is implied by round-by-round soundness.

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 1 (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. We refer to the IOPP in Theorem 1 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 that the rate shrinks by a factor of $2/k$ with each recursion: 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, 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 $O(\log\log d)$, and query complexity $O(\log\log d)$. 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 settings, replacing FRI with STIR leaves the prover time essentially unchanged.

We give concrete examples in Table 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 6 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 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.

[2] A polynomial IOP is an IOP where both honest and malicious provers send bounded-degree polynomials as their oracle messages.

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 $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$ 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 2 (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*

$$\Pr\left[\Delta(\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]) \leq \delta\right] > \mathsf{err}^\star\left(d^*, \rho, \delta, m \cdot (d+1) - \sum d_i\right) \ ,$$

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*

$$\forall\, i \in [m],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i],\ f_i(S) = u(S) \ .$$

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.

2 Techniques

We outline the main ideas behind our results. In Section 2.1, we present the basic properties of STIR and explain how these lead to improved query complexity. In Section 2.2, we describe and analyze a single iteration of STIR. In Section 2.3, we describe a technique to do batch degree correction for functions of different degrees.

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 (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 $|\mathcal{L}'| = n/2$.

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, the improvement in rate is the crucial feature in STIR that facilitates smaller query complexity, as we discuss later in this section.

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.

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

[4] We index the repetitions starting from 0 rather than 1 for notational convenience.

[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 3.1 for a precise definition.

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.

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 $t_i := \frac{\lambda}{-\log\rho_i}$.

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 4.4 for a proof of these properties):

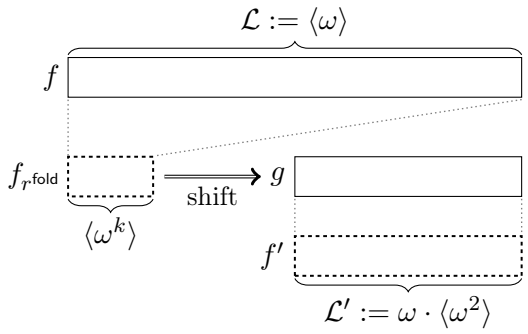


Figure 1: The basic structure of STIR: given r^{fold}, the function f virtually defines $f_{r^{\mathsf{fold}}} := \mathsf{Fold}(f, r^{\mathsf{fold}})$, which is (virtually) evaluated over $\langle \omega^k \rangle$. The prover then sends g evaluated over the larger domain $\mathcal{L}'$. Finally, the function $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ is virtually defined by quotienting g.

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 $1 - \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$ 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 4.2 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 $r^{\mathsf{fold}} \leftarrow \mathbb{F}$.

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 $\mathsf{Fold}(f, r^{\mathsf{fold}})$ to a polynomial of degree less than d/k.

[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.

3. *Out-of-domain sample:* The verifier samples and sends $r^{\mathsf{out}} \leftarrow \mathbb{F} \setminus \mathcal{L}'$.

4. *Out-of-domain reply:* The prover sends a field element $\beta \in \mathbb{F}$. In the honest case, $\beta := \hat{g}(r^{\mathsf{out}})$.

5. *Shift queries:* The verifier, for every $i \in [t]$, samples $r_i^{\mathsf{shift}} \leftarrow \langle \omega^k \rangle$ and obtains $y_i := f_{r^{\mathsf{fold}}}(r_i^{\mathsf{shift}})$ by querying the (virtual) oracle $f_{r^{\mathsf{fold}}}$, where $f_{r^{\mathsf{fold}}} := \mathsf{Fold}(f, r^{\mathsf{fold}})$.

The next function f' is defined as $f' := \mathsf{Quotient}(g, \mathcal{G}, p)$ where $\mathcal{G} := \{r^{\mathsf{out}}, r_1^{\mathsf{shift}}, \ldots, r_t^{\mathsf{shift}}\}$ and $p \colon \mathcal{G} \to \mathbb{F}$ is the function such that $p(r^{\mathsf{out}}) = \beta$ and $p(r_i^{\mathsf{shift}}) = y_i$. Observe that the verifier has virtual oracle access to f' through its oracle access to g.

In Figure 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 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 1. *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 $f_{r^{\mathsf{fold}}} := \mathsf{Fold}(f, r^{\mathsf{fold}})$ is δ-far from low degree. Specifically, by the properties of the folding function, $\Delta(f_{r^{\mathsf{fold}}}, \mathsf{RS}[\mathbb{F}, \langle \omega^k \rangle, d/k]) \geq \delta$ with probability at least $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$.

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 $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$ 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 $\hat{u}(r^{\mathsf{out}}) = \beta$, 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 $\ell = \mathrm{poly}(|\mathcal{L}'|) = \mathrm{poly}(|\mathcal{L}|)$, 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 $O(\ell^2 \cdot d/(k \cdot |\mathbb{F}|)) = \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$.

If Item 1 and Item 2 both hold, which happens with probability $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$, 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 $p(r^{\mathsf{out}}) = \beta$.

- If there exists a codeword u as in Item 2, then by Item 1 the polynomial $\hat{u}$ agrees with $\mathsf{Fold}(f, r^{\mathsf{fold}})$ on at most a $1 - \delta$ fraction of the domain. Thus, the probability that none of the t samples r_i^{shift} hits such a location is at most $(1-\delta)^t$. If a point r_i^{shift} is chosen such that $f(r_i^{\mathsf{shift}}) \neq \hat{u}(r_i^{\mathsf{shift}})$, then there is no polynomial $(1 - \sqrt{\rho'})$-close to g that simultaneously agrees with f on r_i^{shift} 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 5.2 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 2.3.

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 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 7).

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 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 2) 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 1. $\mathsf{deg}(\hat{f}_i) < d^* - e + i$ *for every* i.

Proof sketch. We prove the claim via (reverse) induction on i. The base case is immediate since $\mathsf{deg}(\hat{f}_e) < d^*$. Assuming that $\mathsf{deg}(\hat{f}_{i+1}) < d^* - e + i + 1$, we show that $\mathsf{deg}(\hat{f}_i) < d^* - e + i$. Consider

the polynomial $\hat{p}(X) := X \cdot \hat{f}_i(X)$ and observe that, since $\mathsf{deg}(\hat{f}_i) < d^*$, it holds that $\mathsf{deg}(\hat{p}) < d^*+1$. 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 $\mathsf{deg}(\hat{p}), \mathsf{deg}(\hat{f}_{i+1}) < d^* + 1$, it follows that $\hat{p}$ and $\hat{f}_{i+1}$ are identical. In particular, $\mathsf{deg}(\hat{p}) = \mathsf{deg}(\hat{f}_{i+1}) < d^* - e + i + 1$. Recalling that $\hat{p}(X) := X \cdot \hat{f}_i(X)$, it follows that $\mathsf{deg}(\hat{f}_i) < d^* - e + i$. $\square$

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 $f_1, \ldots, f_m$.

3 Preliminaries

We define objects and state results that we use in this paper. We use the following notation:

- 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 $\mathcal{S} \subseteq \mathbb{F}^{\mathcal{L}}$, $\Delta(f,\mathcal{S}) := \min_{h\in\mathcal{S}} \Delta(f,h)$.
- 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 interactive (oracle) algorithms $\mathbf{A}$ and $\mathbf{B}$, we denote by $\langle \mathbf{A}(a), \mathbf{B}(b)\rangle(c)$ the random variable describing the output of $\mathbf{B}$ following the interaction between $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}$ is given private input a, $\mathbf{B}$ is given private input b, and both parties are given joint input c.
- 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 $\mathcal{R}$.

3.1 Interactive oracle proofs of proximity and their polynomial variant

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

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 $\mathbf{P}$ is an interactive algorithm and $\mathbf{V}$ is an interactive oracle algorithm. We say that IOP is a *public-coin IOP* for a relation $\mathcal{R}$ 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 $\tilde{\mathbf{P}}$,

$$\Pr_{\alpha_1,\ldots,\alpha_{\mathsf{k}}}\left[\mathbf{V}^{\mathbb{y},\pi_1,\ldots,\pi_{\mathsf{k}}}(\mathbb{x},\alpha_1,\ldots,\alpha_{\mathsf{k}}) = 1 \;\middle|\; \begin{array}{r} \pi_1 \leftarrow \tilde{\mathbf{P}}(\alpha_1) \\ \vdots \\ \pi_{\mathsf{k}} \leftarrow \tilde{\mathbf{P}}(\alpha_1,\ldots,\alpha_{\mathsf{k}}) \end{array}\right] \leq \beta(\mathbb{x},\mathbb{y}) \ .$$

  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).

IOPs. An IOP is an IOPP where $\mathbb{y}$ is the empty string (i.e., for a relation $\mathcal{R} = \{(\mathbb{x}, \perp, \mathbb{w})\}$, in which case we generally omit $\perp$ which results in $\mathcal{R}$ being a binary relation).

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

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

State function. 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 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.

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

then $((\mathbb{x}, \mathbb{y}), \mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})) \in \mathcal{R}$.

As with standard soundness, we write ε_i as a function of proximity when appropriate. If et is unbounded, then we omit the word "knowledge" and say that the IOPP has *round-by-round soundness*.

Polynomial IOPPs. 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 field $\mathbb{F}$. In more detail, for every round i there is a prescribed list of m_i degrees $(d_{i,j})_{j \in [m_i]})$ where $d_{i,j} \in \mathbb{N}$. In round i, the prover (both honest and malicious) outputs m_i polynomials by specifying their coefficients, where the j-th polynomial $\hat{f}_{i,j} \in \mathbb{F}^{\leq d_{i,j}}[X]$ has degree at most $d_{i,j}$. The verifier is then given as a message $(\hat{f}_{i,j}(\mathbb{F}))_{j \in m_i}$ where $\hat{f}_{i,j}(\mathbb{F})$ is the evaluation of $\hat{f}_{i,j}$ over the entire field $\mathbb{F}$.

Completeness and soundness for a poly-IOPP are similar to that of standard IOPPs, except in both cases the prover is restricted to sending polynomials as above. Round-by-round knowledge soundness is adapted similarly (where the state function and extractor are given the polynomial coefficients as the prover message). A poly-IOPP has the same parameters as an IOPP, except that, rather than counting the proof length, we count the number of functions:

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

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 $\hat{f}_{i,j}$.

We also use polynomial IOPs (poly-IOP), which are defined similarly with respect to IOPs.

3.2 The Reed–Solomon code

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

Definition 3.2. *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:*

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

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

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).

Definition 3.3. *For a Reed–Solomon code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$, parameter $\delta \in [0, 1]$, and $f \colon \mathcal{L} \to \mathbb{F}$, $\mathsf{List}(f, d, \delta)$ denotes the list of codewords in $\mathcal{C}$ within relative Hamming distance at most δ from f. We say that $\mathcal{C}$ is (δ, ℓ)-***list decodable*** *if $|\mathsf{List}(f, d, \delta)| \leq \ell$ for every f.*

The Johnson bound bounds the list size of the Reed–Solomon code:

Theorem 3.4 (Johnson bound)**.** *The Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is $(1 - \sqrt{\rho} - \eta, 1/(2\eta\sqrt{\rho}))$-list decodable for every $\eta \in (0, 1 - \sqrt{\rho})$, where $\rho := d/|\mathcal{L}|$ is the rate of the code.*

4 Tools for Reed–Solomon codes

We describe tools for Reed–Solomon codes that we use in this paper.

- In Section 4.1 we describe a theorem of [BCIKS20] showing that taking a random linear combination is a good proximity generator for Reed–Solomon codes.
- In Section 4.2 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 4.3 we describe "out-of-domain sampling", a method to reduce the Reed–Solomon list-decoding size of a given function.
- In Section 4.4 we describe how to "fold" a univariate function and show that this preserves the function's distance to the Reed–Solomon code.
- In Section 4.5 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.

4.1 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 4.1 ([BCIKS20])**.** *Let $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ be a Reed–Solomon code with rate $\rho := d/|\mathcal{L}|$, and let $\mathsf{B}^\star(\rho) := \sqrt{\rho}$. For every $\delta \in (0, 1 - \mathsf{B}^\star(\rho))$ and functions $f_1 \ldots, f_m \colon \mathcal{L} \to \mathbb{F}$, if*

$$\Pr_{r \leftarrow \mathbb{F}} \left[\Delta \left(\sum_{j=1}^{m} r^{j-1} \cdot f_j,\ \mathsf{RS}[\mathbb{F}, \mathcal{L}, d] \right) \leq \delta \right] > \mathsf{err}^\star(d, \rho, \delta, m) \ ,$$

then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$, and

$$\forall\, i \in [m],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d],\ f_i(S) = u(S) \ .$$

Above, $\mathsf{err}^\star(d, \rho, \delta, m)$ is defined as follows:

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

$$\mathsf{err}^\star(d, \rho, \delta, m) := \frac{(m-1) \cdot d}{\rho \cdot |\mathbb{F}|} \ .$$

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

$$\mathsf{err}^\star(d, \rho, \delta, m) := \frac{(m-1) \cdot d^2}{|\mathbb{F}| \cdot \left(2 \cdot \min\left\{1 - \sqrt{\rho} - \delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7} \ .$$

4.2 Univariate function quotienting

We define the *quotient* of a univariate function.

Definition 4.2. *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 $\hat{V}_S \in \mathbb{F}^{<|S|}[X]$ 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 4.3. *Let $\hat{f} \in \mathbb{F}^{<d}[X]$ be a polynomial and $S \subseteq \mathbb{F}$ be a set, and let $\hat{V}_S \in \mathbb{F}^{<|S|}[X]$ 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 4.4. *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 $u \in \mathsf{List}(f, d, \delta)$ 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 $T := \{x \in S : \hat{\mathsf{Ans}}(x) \neq f(x)\}$.

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 Definition 4.2. 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 Definition 4.2 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. □

4.3 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 4.5. *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*

$$\Pr_{r_1,\ldots,r_s \leftarrow \mathbb{F}\setminus\mathcal{L}} \left[\exists \text{ distinct } u, u' \in \mathsf{List}(f,d,\delta) : \forall\, i \in [s],\ \hat{u}(r_i) = \hat{u}'(r_i)\right] \leq \binom{\ell}{2} \cdot \left(\frac{d-1}{|\mathbb{F}| - |\mathcal{L}|}\right)^s \leq \frac{d^s \cdot \ell^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}|)^s} \ .$$

Proof. Fix two distinct codewords $u, u' \in \mathsf{List}(f,d,\delta)$. 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 $\binom{\ell}{2} \cdot \left(\frac{d-1}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$. □

4.4 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 polynonmials into bivariate polynomials.

Fact 4.6 ([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 $\mathsf{deg}_X(\hat{Q}) = \left\lfloor \mathsf{deg}(\hat{f})/\mathsf{deg}(\hat{q}) \right\rfloor$ and $\mathsf{deg}_Y(\hat{Q}) < \mathsf{deg}(\hat{q})$ 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 $\mathsf{deg}(\hat{f}) < t \cdot \mathsf{deg}(\hat{q})$ then $\mathsf{deg}_X(\hat{Q}) < t$.*
- *For every $\hat{Q} \in \mathbb{F}[X,Y]$ with $\mathsf{deg}_X(\hat{Q}) < t$ and $\mathsf{deg}_Y(\hat{Q}) < \mathsf{deg}(\hat{q})$, the polynomial $\hat{f}(Z) := \hat{Q}(\hat{q}(Z), Z)$ has degree $\mathsf{deg}(\hat{f}) < t \cdot \mathsf{deg}(\hat{q})$.*

We define the folding of a polynomial and then the folding of a function.

Definition 4.7. *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 Fact 4.6 with $\hat{q}(X) := X^k$. Then $\mathsf{PolyFold}(\hat{f}, k, r)(X) := \hat{Q}(X, r)$.*

Definition 4.8. *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 4.9. *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 $\delta \in (0, \min\{\Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]), 1 - \mathsf{B}^\star(\rho)\})$, letting $\rho := d/|\mathcal{L}|$,*

$$\Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, r^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \leq \delta\right] \leq \mathsf{err}^\star(d/k, \rho, \delta, k) \ .$$

Above, $\mathsf{B}^\star$ and $\mathsf{err}^\star$ are the proximity bound and error (respectively) described in Section 4.1.

Proof. Suppose towards contradiction that

$$\Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, r^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \leq \delta\right] > \mathsf{err}^\star(d/k, \rho, \delta, k) \ .$$

Letting $\hat{p}_x$ be defined from f as in Definition 4.8, 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

$$\mathsf{Fold}(f, k, \alpha)(x) = \hat{p}_x(\alpha) = \sum_{j=0}^{k-1} c_j(x) \cdot \alpha^j = \mathsf{Combine}(\alpha, (c_0, \ldots, c_{k-1})) \ .$$

Therefore, we get that

$$\begin{aligned}
&\Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Combine}(r^{\mathsf{fold}}, (c_0, \ldots, c_{k-1})), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \leq \delta\right] \\
&= \Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f, k, r^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}^k, d/k]) \leq \delta\right] \\
&> \mathsf{err}^\star(d/k, \rho, \delta, k) \ .
\end{aligned}$$

By Theorem 4.1, 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 $\mathsf{deg}_X(\hat{Q}) < d/k$ and $\mathsf{deg}_Y(\hat{Q}) < k$.

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 $\delta < \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d])$. □

4.5 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 4.10. *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 4.11. *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 4.12. *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 $\min\{1 - \mathsf{B}^\star(\rho), 1 - \rho - 1/|\mathcal{L}|\}$.

Lemma 4.13. *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 $\delta \in (0, \min\{1 - \mathsf{B}^\star(\rho), 1 - \rho - 1/|\mathcal{L}|\})$ be a distance parameter, where $\rho := d^*/|\mathcal{L}|$. If*

$$\Pr_{r \leftarrow \mathbb{F}} [\Delta(\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]) \leq \delta] > \mathsf{err}^\star \left(d^*, \rho, \delta, m \cdot (d^* + 1) - \sum_{i=1}^{m} d_i \right) ,$$

then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$, and

$$\forall\, i \in [m],\ \exists\, u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i],\ f_i(S) = u(S) .$$

Note that this implies that $\Delta(f_i, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]) \leq \delta$ for every i. Above, $\mathsf{B}^\star$ and $\mathsf{err}^\star$ are the proximity bound and error (respectively) described in Section 4.1.

Proof. By Definition 4.11, for every r,

$$\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)) = \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{\ell=0}^{d^*-d_i} (r \cdot x)^\ell \right) \quad .$$

Then, since $r_1 = 1$ and $r_i := r^{i-1+\sum_{j<i}(d^*-d_j)}$:

$$\begin{aligned}
&\Pr_{r \leftarrow \mathbb{F}} \left[\Delta \left(\sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{\ell=0}^{d^*-d_i} (r \cdot x)^\ell \right), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*] \right) \leq \delta \right] \\
&= \Pr_{r \leftarrow \mathbb{F}} [\Delta(\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d^*]) \leq \delta] \\
&> \mathsf{err}^\star \left(d^*, \rho, \delta, m \cdot (d^* + 1) - \sum_{i=1}^{m} d_i \right) \quad .
\end{aligned}$$

By Theorem 4.1, 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 $\mathsf{deg}(\hat{p}_{i,j}) < d_i + j$. 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 $\mathsf{deg}(\hat{p}_{i,d^*-d_i}) < d^* = d_i + d^* - d_i$. For $0 \leq j < d^* - d_i$ suppose that $\mathsf{deg}(\hat{p}_{i,j+1}) < d_i + j + 1$. We show that $\mathsf{deg}(\hat{p}_{i,j}) < d_i + j$. Consider the polynomial $\hat{q}(X) := X \cdot \hat{p}_{i,j}(X)$. Since $\mathsf{deg}(\hat{p}_{i,j}) < d$, it follows that $\mathsf{deg}(\hat{q}) < d^* + 1$. Observe that for every $x \in S$,

$$\hat{q}(x) = x \cdot \hat{p}_{i,j}(x) = x^{j+1} \cdot f_i(x) = \hat{p}_{i,j+1}(x) \quad .$$

The polynomials $\hat{q}$ and $\hat{p}_{i,j+1}$ have degree less than d^*+1, and agree on $|S| \geq (1-\delta) \cdot |\mathcal{L}| > (\rho + 1/|\mathcal{L}|) \cdot |\mathcal{L}| = d^* + 1$ points. They are therefore identical and, in particular, $\mathsf{deg}(\hat{q}) = \mathsf{deg}(\hat{p}_{i,j+1}) < d^* + j + 1$. Recalling that $\hat{q}(X) := X \cdot \hat{p}_{i,j}(X)$ we conclude that $\mathsf{deg}(\hat{p}_{i,j}) < d_i + j$. □

5 STIR

We describe STIR, an interactive oracle proof of proximity for nice Reed–Solomon codes.

- In Section 5.1 we describe the construction and analyze its complexity parameters.
- In Section 5.2 we prove round-by-round soundness of STIR.
- In Section 5.3 we give recommended settings of parameters for STIR, including a numeric example.

Theorem 5.1. *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: $O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{\log 1/\rho}\right)\right)$.

5.1 Construction

We describe STIR and analyze its complexity parameters.

Construction 5.2. 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 $s \in \mathbb{N}$.

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 $r_0^{\mathsf{fold}} \leftarrow \mathbb{F}$.
- **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 $\hat{g}_i := \mathsf{PolyFold}(\hat{f}_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}})$ over $\mathcal{L}_i$.

[7] If, additionally, $\mathcal{L}_i \cap \mathcal{L}_{i+1} = \emptyset$ for every i, then the protocol can be made more efficient. See Remark 5.3.

2. **Out-of-domain samples:** The verifier sends $r^{\mathsf{out}}_{i,1}, \ldots, r^{\mathsf{out}}_{i,s} \leftarrow \mathbb{F} \setminus \mathcal{L}_i$.
3. **Out-of-domain reply:** The prover sends field elements $\beta_{i,1}, \ldots, \beta_{i,s} \in \mathbb{F}$. In the honest case, $\beta_{i,j} := \hat{g}_i(r^{\mathsf{out}}_{i,j})$.
4. **STIR message:** The verifier sends $r^{\mathsf{fold}}_i, r^{\mathsf{comb}}_i \leftarrow \mathbb{F}$ and $r^{\mathsf{shift}}_{i,1}, \ldots, r^{\mathsf{shift}}_{i,t_{i-1}} \leftarrow \mathcal{L}^k_{i-1}$.
5. **Define next polynomial and send hole fills:** The prover sends oracle message $\mathsf{Fill}_i \colon \{r^{\mathsf{shift}}_{i,1}, \ldots, r^{\mathsf{shift}}_{i,t_{i-1}}\} \cap \mathcal{L}_i \to \mathbb{F}$. In the honest case, the prover defines $\mathcal{G}_i := \{r^{\mathsf{out}}_{i,1}, \ldots, r^{\mathsf{out}}_{i,s}, r^{\mathsf{shift}}_{i,1}, \ldots, r^{\mathsf{shift}}_{i,t_{i-1}}\}$, $\hat{g}'_i := \mathsf{PolyQuotient}(\hat{g}_i, \mathcal{G}_i)$, and $\mathsf{Fill}_i(r^{\mathsf{shift}}_{i,j}) := \hat{g}'_i(r^{\mathsf{shift}}_{i,j})$ (if $r^{\mathsf{shift}}_{i,j} \in \mathcal{L}_i$).

   Additionally, the honest prover defines the degree-corrected polynomial $\hat{f}_i \in \mathbb{F}^{<d_i}[X]$ as follows:

$$\hat{f}_i := \mathsf{DegCor}(d_i, r^{\mathsf{comb}}_i, \hat{g}'_i, d_i - |\mathcal{G}_i|) \ .$$

   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, $\hat{p} := \mathsf{Fold}(\hat{f}_M, k_M, r^{\mathsf{fold}}_M)$.
- **Verifier decision phase**:
  1. **Main loop:** For $i = 1, \ldots, M$:
     (a) For every $j \in [t_{i-1}]$, query $\mathsf{Fold}(f_{i-1}, k_{i-1}, r^{\mathsf{fold}}_{i-1})$ at $r^{\mathsf{shift}}_{i,j}$. This involves querying f_{i-1} at all k_{i-1} points $x \in \mathcal{L}_{i-1}$ with $x^{k_{i-1}} = r^{\mathsf{shift}}_{i,j}$.
     (b) Define $\mathcal{G}_i := \{r^{\mathsf{out}}_{i,1}, \ldots, r^{\mathsf{out}}_{i,s}, r^{\mathsf{shift}}_{i,1}, \ldots, r^{\mathsf{shift}}_{i,t_{i-1}}\}$, and let $\mathsf{Ans}_i \colon \mathcal{G}_i \to \mathbb{F}$ be the function where $\mathsf{Ans}_i(r^{\mathsf{out}}_{i,j}) = \beta_{i,j}$ and $\mathsf{Ans}_i(r^{\mathsf{shift}}_{i,j}) = \mathsf{Fold}(f_{i-1}, k_{i-1}, r^{\mathsf{fold}}_{i-1})(r^{\mathsf{shift}}_{i,j})$. 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:

$$f_i := \mathsf{DegCor}(d_i, r^{\mathsf{comb}}_i, g'_i, d_i - |\mathcal{G}_i|) \ .$$

     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 $r^{\mathsf{fin}}_1, \ldots, r^{\mathsf{fin}}_{t_M} \leftarrow \mathcal{L}_M$.
     (b) Check that $\hat{p}(r^{\mathsf{fin}}_j) = \mathsf{Fold}(f_M, k_M, r^{\mathsf{fold}}_M)(r^{\mathsf{fin}}_j)$ 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 5.3. If $\mathcal{L}^{k_{i-1}}_{i-1} \cap \mathcal{L}_i = \emptyset$ 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 Construction 5.2.

- *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 $\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}})$, 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 to 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 $2 \cdot \sum_{i=1}^{M} t_i$.

5.2 Round-by-round soundness

We analyze the round-by-round soundness of STIR.

Lemma 5.4. *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 Construction 5.2, 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*

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1 - \mathsf{B}^\star(\rho_0))$,
- *for every $0 < i \leq M$: $\delta_i \in (0,\ \min\{1 - \rho_i - 1/|\mathcal{L}_i|,\ 1 - \mathsf{B}^\star(\rho_i)\})$, and*
- *for every $0 < i \leq M$: $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]$ is (δ_i, ℓ_i)-list decodable,*

STIR (Construction 5.2) 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:

- $\varepsilon^{\mathsf{fold}} \leq \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0)$.
- $\varepsilon_i^{\mathsf{out}} \leq \frac{d_i^s \cdot \ell_i^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s}$.
- $\varepsilon_i^{\mathsf{shift}} \leq (1 - \delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1} + s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i)$.
- $\varepsilon^{\mathsf{fin}} \leq (1 - \delta_M)^{t_M}$.

Above, $\mathsf{B}^\star$ and $\mathsf{err}^\star$ are the proximity bound and error (respectively) described in Section 4.1.

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

$$\mathsf{tr} := (r_0^{\mathsf{fold}}, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, \mathsf{tr}') \ ,$$

where $\mathsf{tr}_j := (r_j^{\mathsf{out}}, \beta_j, (r_j^{\mathsf{fold}}, r_j^{\mathsf{comb}}, r_{j,1}^{\mathsf{shift}}, \ldots, r_{j,t_j}^{\mathsf{shift}}))$ 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 r_0^{fold}.

   - *State function.* We set $\mathsf{State}(f, r_0^{\mathsf{fold}}) = 1$ if and only if

   $$\Delta(\mathsf{Fold}(f, k_0, r_0^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}^{k_0}, d_0/k_0]) \leq \delta_0 \ .$$

   - *Bounding the error.* Since $\delta_0 < 1 - \mathsf{B}^\star(\rho_0)$, from Lemma 4.9 we obtain that

   $$\begin{aligned}
   \varepsilon^{\mathsf{fold}} &= \Pr_{r_0^{\mathsf{fold}}} \left[\mathsf{State}(f, r_0^{\mathsf{fold}}) = 1 \,\middle|\, \mathsf{State}(f, \emptyset) = 0 \ \right] \\
   &= \Pr_{r_0^{\mathsf{fold}} \leftarrow \mathbb{F}} \left[\Delta(\mathsf{Fold}(f, k_0, r_0^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}^{k_0}, d_0/k_0]) \leq \delta_0 \right] \\
   &< \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0) \ .
   \end{aligned}$$

2. **Bounding $\varepsilon_i^{\mathsf{out}}$.** The transcript so far has the form $\mathsf{tr} := (r_0^{\mathsf{fold}}, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, g_i)$, 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. $\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_i]) \leq \delta_{i-1}$, or
        ii. there exist distinct codewords $u, u' \in \mathsf{List}(g_i, d_i, \delta_i)$ 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

   $$\varepsilon_i^{\mathsf{out}} = \Pr_{r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}}} \left[\mathsf{State}(f, \mathsf{tr} || (r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \ \right] \leq \frac{d_i \cdot \ell_i^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)} \ .$$

   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 $\mathsf{State}(f, \mathsf{tr}) = 0$:

   $$\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_{i-1}/k_{i-1}]) > \delta_{i-1} \ .$$

   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 Lemma 4.5 this probability is at most $\frac{d_i^s \cdot \ell_i^2}{2 \cdot (|\mathbb{F}| - |\mathcal{L}_i|)^s}$.

3. **Bounding $\varepsilon_i^{\mathsf{shift}}$.** The transcript so far has the form $\mathsf{tr} := (r_0^{\mathsf{fold}}, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, g_i, r_i^{\mathsf{out}}, \beta_i)$ and the verifier sends $r_i^{\mathsf{fold}}, r_i^{\mathsf{comb}}, r_1^{\mathsf{shift}}, \ldots, r_{t_{i-1}}^{\mathsf{shift}}$. Deriving f_{i-1} from tr, we define the state function.

   - *State function.* We set $\mathsf{State}(f, \mathsf{tr}||(r_i^{\mathsf{fold}}, r_i^{\mathsf{comb}}, r_1^{\mathsf{shift}}, \ldots, r_{t_{i-1}}^{\mathsf{shift}})) = 1$ if and only if both of the following hold:
     (a) $\Delta(\mathsf{Fold}(f_i, k_i, r_i^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i^{k_i}, d_i/k_i]) \leq \delta_i$ where f_i is defined using f_{i-1}, g_i, $(r_{i,j}^{\mathsf{out}}, \beta_{i,j})_{j\in[s]}$, r_i^{comb}, and $r_{i,1}^{\mathsf{shift}}, \ldots, r_{i,t}^{\mathsf{shift}}$ 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

$$\begin{aligned}\varepsilon_i^{\mathsf{shift}} &= \Pr_{\substack{r_i^{\mathsf{fold}}, r_i^{\mathsf{comb}},\\ r_1^{\mathsf{shift}}, \ldots, r_{t_{i-1}}^{\mathsf{shift}}}} \left[\mathsf{State}(f, \mathsf{tr}||(r_i^{\mathsf{fold}}, r_i^{\mathsf{comb}}, r_1^{\mathsf{shift}}, \ldots, r_{t_{i-1}}^{\mathsf{shift}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \right] \\ &\leq (1-\delta_{i-1})^{t_{i-1}} + \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \ .\end{aligned}$$

   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 5.5. *With probability $1-(1-\delta_{i-1})^{t_{i-1}}$ over the choice of $r_{i,1}^{\mathsf{shift}}, \ldots, r_{i,t_{i-1}}^{\mathsf{shift}}$ for every $u \in \mathsf{List}(g_i, d_i, \delta_i)$ 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 $u \in \mathsf{List}(g_i, d_i, \delta_i)$ such that $\hat{u}(r_{i,j}^{\mathsf{out}}) = \beta_{i,j}$ for every $j \in [s]$. If no such codeword exists, then the claim holds trivially since $\mathsf{Ans}_i(r_{i,j}^{\mathsf{out}}) = \beta_{i,j}$ for some j. We are left to analyze the case where there is exactly one codeword $u \in \mathsf{List}(g_i, d_i, \delta_i)$ for which $\hat{u}(r_{i,j}^{\mathsf{out}}) = \beta_{i,j}$ for every j.
   Since $\mathsf{State}(f_0, \mathsf{tr}) = 0$ we have that

$$\Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}}), \hat{u}(\mathcal{L}_{i-1}^{k_{i-1}})) \geq \Delta(\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_{i-1}^{k_{i-1}}, d_i]) > \delta_{i-1} \ .$$

   Thus, for every j the probability that $\hat{u}(r_{i,j}^{\mathsf{shift}}) \neq \mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}})(r_{i,j}^{\mathsf{shift}}) = \mathsf{Ans}_i(r_{i,j}^{\mathsf{shift}})$ 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 Claim 5.5 occurs. Then by Lemma 4.4:

$$\Delta\left(g_i', \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i - |\mathcal{G}_i|]\right) + \frac{|\{x \in \mathcal{G}_i : g_i(x) \neq \mathsf{Ans}_i(x)\}|}{|\mathcal{L}_i|} > \delta_i \ .$$

   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 $\delta_i < \min\{1 - \mathsf{B}^\star(\rho_i), 1 - \rho_i - 1/|\mathcal{L}_i|\}$ and using Lemma 4.13 we deduce that

$$\Pr_{r_i^{\mathsf{comb}} \leftarrow \mathbb{F}} \left[\Delta\left(\mathsf{DegCor}(d_i, r_i^{\mathsf{comb}}, g_i', d_i - |\mathcal{G}_i|), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]\right) \leq \delta_i \right] \leq \mathsf{err}^\star(d_i, \rho_i, \delta_i, t_{i-1}+s) \ .$$

Finally, observe that $f_i := \mathsf{DegCor}(d_i, r_i^{\mathsf{comb}}, g_i', d_i - |\mathcal{G}_i|)$ and that if $\Delta\left(f_i, \mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d_i]\right) \leq \delta_i$, then by Lemma 4.9:

$$\Pr_{r_i^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f_i, k_i, r_i^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_i^{k_i}, d_i/k_i]) \leq \delta_i\right] \leq \mathsf{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \ .$$

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 $\varepsilon^{\mathsf{fin}}$:** At this stage the transcript holds the form $\mathsf{tr} := (r_0^{\mathsf{fold}}, \mathsf{tr}_1, \ldots, \mathsf{tr}_M, \hat{p})$. The verifier chooses points $(r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}})$. Deriving f_M from the transcript, we define the state function.

   - *State function.* We set $\mathsf{State}(f, \mathsf{tr}||(r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}})) = 1$ if and only if both of the following hold:
     (a) $\hat{p}(r_j^{\mathsf{fin}}) = \mathsf{Fold}(f_M, k, r_M^{\mathsf{fold}})(r_j^{\mathsf{fin}})$ 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

   $$\varepsilon^{\mathsf{fin}} = \Pr_{r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}}}\left[\mathsf{State}(f, \mathsf{tr}||(r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}})) = 1 \,\middle|\, \mathsf{State}(f, \mathsf{tr}) = 0 \ \right] \leq (1-\delta_M)^{t_M} \ .$$

   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:

   $$\Delta(\mathsf{Fold}(f_M, k_M, r_M^{\mathsf{fold}}), \hat{p}(\mathcal{L}_M^{k_M})) \geq \Delta(\mathsf{Fold}(f_M, k_M, r_M^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F}, \mathcal{L}_M^{k_M}, d_M]) > \delta_M \ .$$

   Thus, for each j the probability that $\hat{p}(r_j^{\mathsf{fin}}) = \mathsf{Fold}(f_M, k_M, r_M^{\mathsf{fold}})(r_j^{\mathsf{fin}})$ 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.

□

5.3 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 Appendix C.

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 $d_{\mathsf{stop}} \in \mathbb{N}$ 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 Construction 5.2 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 Appendix C.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 5.6 ([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*

$$\Pr_{r \leftarrow \mathbb{F}}\left[\Delta\left(\mathcal{C}, \mathsf{Combine}(r, (f_1, \ldots, f_m))\right) \leq \delta\right] > \mathsf{err}^\star(d, \rho, \delta, m) := \frac{(m-1)^{c_2} \cdot d^{c_2}}{\eta^{c_1} \cdot \rho^{c_1+c_2} \cdot |\mathbb{F}|} .$$

  then there exists $S \subseteq \mathcal{L}$ with $|S| \geq (1-\delta) \cdot |\mathcal{L}|$, and

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

- *$\mathcal{C}$ is (δ, ℓ)–list decodable for $\ell \leq \left(\frac{d}{\rho \cdot \eta}\right)^{c_3}$.*

[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 Remark 5.3.

[9] Note that this bound is not tight.

The ingredients are identical as in the provable setting. The parameters in Construction 5.2 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

$$\eta_i := \max\left\{\frac{2\cdot\rho_i}{d_i}, \frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda \cdot d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s}\right)^{\frac{1}{2\cdot c_3}}, \left(\frac{2^{\lambda+1}\cdot d_i^{c_2}}{\rho_i^{c_1+c_2}\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+s-1)^{c_2}+\left(\frac{k-1}{k}\right)^{c_2}\right)\right)^{1/c_1}\right\} .$$

Assuming Conjecture 5.6, 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 Appendix C.2 for full derivations given this parameter setting.

6 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.

6.1 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 Appendix B with the compiler in Section 7. For now, the sizes that we report are obtained via a Python script.

6.2 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 $|\mathcal{L}| = d/\rho$.

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.

[10] We arbitrarily selected $\mathbb{F} = \mathbb{F}_p$ with $p = 2^{64} \cdot 259536638529657107390708680683681617371 + 1$.

Repetitions. The number of repetitions in FRI and in STIR to achieve round-by-round soundness error $2^{-\lambda_{\mathsf{prot}}}$ was selected by assuming Conjecture 5.6 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 Appendix B, 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).

6.3 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 6.2. 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.

6.4 Results

We discuss the results of our experiments with the *native* configuration. Figure 2 graphically compares STIR and FRI for rate $1/2$ and varying degrees. Table 2 contains all experimental data for the native configuration. Additional experimental results are in Appendix A. We discuss each metric individually, arbitrarily focusing on the case of degree $d = 2^{24}$ and rate $\rho = 1/2$.

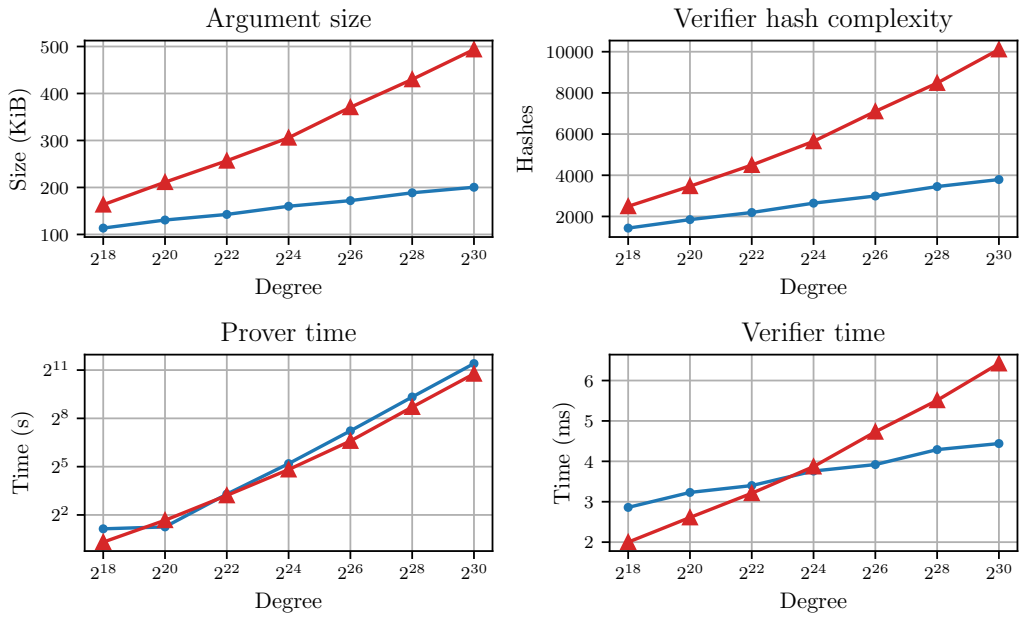


Figure 2: Comparison of FRI and STIR for $\rho = 1/2$. FRI: ▲, STIR: ●. Lower is better.

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 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×.

[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.

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. 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, 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 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 3: Comparison of STIR and FRI on different rates.

ρ \ d	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
	Argument size (KiB, $\frac{\text{FRI}}{\text{STIR}}$)						
$1/2$	$\frac{163}{114} \approx 1.44\times$	$\frac{211}{131} \approx 1.62\times$	$\frac{257}{143} \approx 1.8\times$	$\frac{306}{160} \approx 1.91\times$	$\frac{371}{172} \approx 2.15\times$	$\frac{430}{189} \approx 2.28\times$	$\frac{494}{200} \approx 2.46\times$
$1/4$	$\frac{99}{73} \approx 1.34\times$	$\frac{129}{87} \approx 1.48\times$	$\frac{154}{94} \approx 1.63\times$	$\frac{177}{107} \approx 1.66\times$	$\frac{211}{114} \approx 1.84\times$	$\frac{249}{128} \approx 1.95\times$	$\frac{277}{136} \approx 2.04\times$
$1/8$	$\frac{76}{58} \approx 1.32\times$	$\frac{96}{69} \approx 1.39\times$	$\frac{118}{75} \approx 1.57\times$	$\frac{134}{86} \approx 1.55\times$	$\frac{157}{93} \approx 1.7\times$	$\frac{184}{104} \approx 1.77\times$	$\frac{204}{110} \approx 1.85\times$
$1/16$	$\frac{62}{50} \approx 1.25\times$	$\frac{77}{61} \approx 1.27\times$	$\frac{95}{66} \approx 1.44\times$	$\frac{107}{76} \approx 1.41\times$	$\frac{127}{82} \approx 1.56\times$	$\frac{147}{92} \approx 1.6\times$	-
	Verifier time (ms, $\frac{\text{FRI}}{\text{STIR}}$)						
$1/2$	$\frac{2.0}{2.9} \approx 0.7\times$	$\frac{2.6}{3.2} \approx 0.81\times$	$\frac{3.2}{3.4} \approx 0.94\times$	$\frac{3.9}{3.8} \approx 1.03\times$	$\frac{4.7}{3.9} \approx 1.21\times$	$\frac{5.5}{4.3} \approx 1.28\times$	$\frac{6.4}{4.4} \approx 1.45\times$
$1/4$	$\frac{1.2}{1.7} \approx 0.74\times$	$\frac{1.6}{2.0} \approx 0.8\times$	$\frac{1.9}{2.1} \approx 0.92\times$	$\frac{2.3}{2.4} \approx 0.97\times$	$\frac{2.7}{2.5} \approx 1.09\times$	$\frac{3.2}{2.8} \approx 1.17\times$	$\frac{3.7}{2.9} \approx 1.27\times$
$1/8$	$\frac{1.0}{1.2} \approx 0.78\times$	$\frac{1.2}{1.5} \approx 0.79\times$	$\frac{1.5}{1.6} \approx 0.93\times$	$\frac{1.8}{1.9} \approx 0.95\times$	$\frac{2.1}{2.0} \approx 1.06\times$	$\frac{2.4}{2.2} \approx 1.11\times$	$\frac{2.8}{2.3} \approx 1.22\times$
$1/16$	$\frac{0.8}{1.0} \approx 0.79\times$	$\frac{1.0}{1.3} \approx 0.76\times$	$\frac{1.2}{1.4} \approx 0.89\times$	$\frac{1.4}{1.6} \approx 0.89\times$	$\frac{1.7}{1.7} \approx 1.01\times$	$\frac{2.0}{1.9} \approx 1.04\times$	-
	Verifier hashes ($\frac{\text{FRI}}{\text{STIR}}$)						
$1/2$	$\frac{2490}{1434} \approx 1.74\times$	$\frac{3466}{1846} \approx 1.88\times$	$\frac{4494}{2191} \approx 2.05\times$	$\frac{5647}{2645} \approx 2.13\times$	$\frac{7100}{2992} \approx 2.37\times$	$\frac{8479}{3451} \approx 2.46\times$	$\frac{10107}{3792} \approx 2.67\times$
$1/4$	$\frac{1658}{1020} \approx 1.63\times$	$\frac{2270}{1329} \approx 1.71\times$	$\frac{2821}{1521} \approx 1.85\times$	$\frac{3459}{1849} \approx 1.87\times$	$\frac{4220}{2050} \approx 2.06\times$	$\frac{5072}{2401} \approx 2.11\times$	$\frac{5885}{2622} \approx 2.24\times$
$1/8$	$\frac{1374}{843} \approx 1.63\times$	$\frac{1801}{1098} \approx 1.64\times$	$\frac{2258}{1256} \approx 1.8\times$	$\frac{2720}{1534} \approx 1.77\times$	$\frac{3271}{1697} \approx 1.93\times$	$\frac{3879}{2010} \approx 1.93\times$	$\frac{4455}{2172} \approx 2.05\times$
$1/16$	$\frac{1185}{765} \approx 1.55\times$	$\frac{1518}{1014} \approx 1.5\times$	$\frac{1898}{1147} \approx 1.65\times$	$\frac{2233}{1376} \approx 1.62\times$	$\frac{2718}{1537} \approx 1.77\times$	$\frac{3166}{1792} \approx 1.77\times$	-
	Prover time (s, $\frac{\text{FRI}}{\text{STIR}}$)						
$1/2$	$\frac{1.2}{2.2} \approx 0.56\times$	$\frac{3.2}{2.4} \approx 1.33\times$	$\frac{9.3}{9.8} \approx 0.95\times$	$\frac{28}{36} \approx 0.77\times$	$\frac{97}{150} \approx 0.65\times$	$\frac{420}{640} \approx 0.65\times$	$\frac{1700}{2700} \approx 0.64\times$
$1/4$	$\frac{2.3}{1.1} \approx 2.11\times$	$\frac{2.7}{3.9} \approx 0.7\times$	$\frac{11}{14} \approx 0.81\times$	$\frac{47}{58} \approx 0.8\times$	$\frac{200}{250} \approx 0.8\times$	$\frac{860}{1100} \approx 0.78\times$	$\frac{3600}{4800} \approx 0.75\times$
$1/8$	$\frac{1.4}{1.9} \approx 0.73\times$	$\frac{5.4}{6.1} \approx 0.89\times$	$\frac{22}{26} \approx 0.86\times$	$\frac{93}{110} \approx 0.85\times$	$\frac{400}{480} \approx 0.83\times$	$\frac{1700}{2100} \approx 0.8\times$	$\frac{7000}{8900} \approx 0.79\times$
$1/16$	$\frac{2.7}{2.9} \approx 0.93\times$	$\frac{10}{11} \approx 0.93\times$	$\frac{44}{48} \approx 0.93\times$	$\frac{190}{210} \approx 0.88\times$	$\frac{780}{930} \approx 0.84\times$	$\frac{3300}{4100} \approx 0.82\times$	-

Table 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.

7 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 7.1. *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* $(\hat{f}_{i,j} \in \mathbb{F}^{<d_{i,j}}[X])_{j\in[m_i]}$ *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* $\max_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_i]}\{d_{i,j}\} \leq d$.
- *A proximity parameter* $\delta \in (0, 1-\max\{\mathsf{B}^\star(\rho), \rho+1/|\mathcal{L}|\})$ *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 $\mathcal{C}$	$\to$ 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	m_{poly}	$\mathsf{l}_{\mathsf{prx}}$	$\mathsf{l}_{\mathsf{prx}} + 2 \cdot \mathsf{q}_{\mathsf{poly},\pi} + m_{\mathsf{poly}} \cdot (\lvert\mathcal{L}\rvert + 2)$
Input queries	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$	$\mathsf{q}_{\mathsf{prx},f}$	$\mathsf{q}_{\mathsf{poly},\mathbb{y}}$
Proof queries	$\mathsf{q}_{\mathsf{poly},\pi}$	$\mathsf{q}_{\mathsf{prx},\pi}$	$m_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\pi} + \mathsf{q}_{\mathsf{poly},\pi}$
Verifier time	$\mathsf{vt}_{\mathsf{poly}}$	$\mathsf{vt}_{\mathsf{prx}}$	$O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{vt}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \mathsf{q}^2_{i,j})$
Extraction time	$\mathsf{et}_{\mathsf{poly}}$	-	$\mathsf{et}_{\mathsf{poly}} + m_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}}$

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* $(\mathbb{x}, \mathbb{y})$*):*

- $\varepsilon^{\mathsf{out}}_i \leq \frac{\sum_{j\in[m_i]} d_{i,j}\cdot\ell^2_{i,j}}{2\cdot|\mathbb{F}|}$.
- $\varepsilon^{\mathsf{piop}}_i \leq \mathsf{err}^{\mathsf{poly}}_i(\mathbb{x}, \mathbb{y})$.
- $\varepsilon^{\mathsf{com}} \leq \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i}(d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right)$.
- $\varepsilon^{\mathsf{prx}}_i \leq \mathsf{err}^{\mathsf{prx}}_i(\delta)$.

Above, $\mathsf{B}^\star$ *and* $\mathsf{err}^\star$ *are the proximity bound and error (respectively) described in Section 4.1.*

7.1 Construction

We describe the transformation from a poly-IOPP to an IOPP, and then discuss its efficiency.

Construction 7.2. We construct an IOPP for $\mathcal{R}$ from the ingredients in Theorem 7.1.

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.

[12] Note that m_{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.

1. **Poly-IOP interaction phase:**

   (a) For $i = 1, \ldots, \mathsf{k}_{\mathsf{poly}}$:

      i. **Poly-IOP prover message:** The prover sends oracle functions $(f_{i,j})_{j\in[m_i]}$ where $f_{i,j}\colon \mathcal{L} \to \mathbb{F}$. In the honest case, the prover computes $(\hat{f}_{i,j})_{j\in[m_i]} := \mathbf{P}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \alpha_1, \ldots, \alpha_{i-1})$, 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 $(y_{i,j})_{j\in[m_i]}$. In the honest case $y_{i,j} := \hat{f}_{i,j}(x_i)$.

      iv. **Poly-IOP verifier message:** The verifier sends $\alpha_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.

2. **Low-degree test interaction phase:**

   (a) **Send query results and prepare for low-degree test:** The prover sends arrays of field elements $(A_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m]}$ where $|A_{i,j}| = \mathsf{q}_{i,j}$, and oracle functions $(w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_i]}$ where $w_{i,j}\colon [\mathsf{q}_{i,j}+1] \to \mathbb{F}$.

   In the honest case the prover simulates the execution of

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

   For every $i \in [\mathsf{k}_{\mathsf{poly}}]$ and $j \in [m_i]$, 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:

   $$\hat{g} := \mathsf{Combine}\left(d, r, (\hat{f}'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_i]}\right) \ ,$$

   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 $\mathbf{V}_{\mathsf{poly}}^{\mathbb{y},(\hat{f}_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_i]}}(\mathbb{x}, \alpha_1, \ldots, \alpha_{\mathsf{k}_{\mathsf{poly}}}) = 1$, 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 $j \in [m_i]$, 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 $j \in [m_i]$:

- 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:

$$\mathsf{Ans}_{i,j}(q) := \begin{cases} A_{i,j}[\phi_{i,j}^{-1}(q)] & q \in \mathcal{Q}_{i,j} \\ y_{i,j} & q = x_i \end{cases} \qquad \mathsf{Fill}_{i,j}(q) := \begin{cases} w_{i,j}(\phi_{i,j}^{-1}(q)) & q \in \mathcal{Q}_{i,j} \\ w_{i,j}(\mathsf{q}_{i,j}+1) & q = x_i \end{cases}$$

- 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

$$g := \mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i \in [\mathsf{k}_{\mathsf{poly}}], j \in [m_i]}\right) \ ,$$

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 $j \in [m_i]$ 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

$$\mathsf{l}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i} |\mathcal{L}| + |S_{i,j}| \leq \mathsf{l}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i} (|\mathcal{L}| + \mathsf{q}_{i,j} + 1) = \mathsf{l}_{\mathsf{prx}} + \mathsf{q}_{\mathsf{poly},\pi} + m_{\mathsf{poly}} \cdot (|\mathcal{L}| + 1) \ .$$

  The prover additionally sends $\sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i} |S_{i,j}| \leq \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i} \mathsf{q}_{i,j} + 1 = \mathsf{q}_{\mathsf{poly},\pi} + m_{\mathsf{poly}}$ field elements. Thus, the total proof length is

$$\mathsf{l}_{\mathsf{prx}} + 2 \cdot \mathsf{q}_{\mathsf{poly},\pi} + m_{\mathsf{poly}} \cdot (|\mathcal{L}| + 2) \ .$$

- *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 $m_{\mathsf{poly}} \cdot \mathsf{q}_{\mathsf{prx},f} + \mathsf{q}_{\mathsf{prx},\pi} + \mathsf{q}_{\mathsf{poly},\pi}$.

- *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 $O(\mathsf{vt}_{\mathsf{poly}} + \mathsf{vt}_{\mathsf{prx}} + \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \mathsf{q}_{i,j}^2)$.

7.2 Round-by-round knowledge soundness

We prove the round-by-round knowledge soundness of the IOPPP in Construction 7.2.

Lemma 7.3. *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* $j \in [m_i]$*,* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{i,j}]$ *is* $(\delta, \ell_{i,j})$*-list decodable. For every* $\delta \in (0, \min\{1-\rho-1/|\mathcal{L}|, 1-\mathsf{B}^\star(\rho)\})$*, the IOPP described in Construction 7.2 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* $O(\mathsf{et}_{\mathsf{poly}} + m_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}})$ *where:*

- $\varepsilon_i^{\mathsf{out}} \leq \frac{\sum_{j\in[m_i]} d_{i,j}\cdot \ell_{i,j}^2}{2\cdot|\mathbb{F}|}$;
- $\varepsilon_i^{\mathsf{piop}} \leq \mathsf{err}_i^{\mathsf{poly}}(\mathbb{x}, \mathbb{y})$;
- $\varepsilon^{\mathsf{com}} \leq \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} \sum_{j=1}^{m_i} (d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right)$;
- $\varepsilon_i^{\mathsf{prx}} \leq \mathsf{err}_i^{\mathsf{prx}}(\delta)$.

Above, $\mathsf{B}^\star$ *and* $\mathsf{err}^\star$ *are the proximity bound and error (respectively) described in Section 4.1.*

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

$$\mathsf{tr} := (((f_{\ell,j})_{j\in[m_i]}, x_\ell, (y_{\ell,j})_{j\in[m_\ell]})_{\ell<i}, (f_{i,j})_{j\in[m_i]}) \ ,$$

   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 $\ell \leq i$ and $j \in [m_i]$, and distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$ such that $\hat{u}_{\ell,j}(x_\ell) = \hat{u}'_{\ell,j}(x_\ell)$.
     (b) There exist codewords $(u_{\ell,j})_{\ell<i, j\in[m_\ell]}$ such that:
       i. $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$,
       ii. $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$, and
       iii. $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell<i}) = 1$.
   - *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

$$\Pr_{x_i}\left[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||x_i) = 1\right] \leq \varepsilon_i^{\mathsf{out}} = \frac{\sum_{j\in[m_i]} d_{i,j}\cdot \ell_{i,j}^2}{2\cdot|\mathbb{F}|} \ .$$

     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 $(u_{\ell,j})_{\ell<i,j\in[m_\ell]}$ such that $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$, $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ and,

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, (\hat{u}_{1,j})_{j\in[m_1]}, \alpha_1, \ldots, (\hat{u}_{i-1,j})_{j\in[m_{i-1}]}, \alpha_{i-1}) = 1 \ .$$

This is precisely stating that Item 1b does not hold. Moreover, by Item 2a, for every $\ell < i$, $j \in [m_i]$, and a pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$ 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 Lemma 4.5 for every fixed $j \in [m_i]$ there exist a pair of distinct codewords $u_{i,j}, u'_{i,j} \in \mathsf{List}(f_{i,j}, d_{i,j}, \delta)$ 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 $\frac{\sum_{j\in[m_i]} d_{i,j}\cdot\ell_{i,j}^2}{2\cdot|\mathbb{F}|}$. As a result, Item 1a holds with probability at most $\frac{\sum_{j\in[m_i]} d_{i,j}\cdot\ell_{i,j}^2}{2\cdot|\mathbb{F}|}$.

2. **Bounding $\varepsilon_i^{\mathsf{piop}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := (((f_{\ell,j})_{j\in[m_\ell]}, x_\ell, (y_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell<i}, (f_{i,j})_{j\in[m_i]}, x_i, (y_{i,j})_{j\in[m_i]}) \ ,$$

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 $\ell \leq i$, $j \in [m_i]$, and a pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$ such that $\hat{u}_{\ell,j}(x_\ell) = \hat{u}'_{\ell,j}(x_\ell)$ or,
  (b) There exist codewords $(u_{\ell,j})_{\ell\leq i, j\in[m_\ell]}$ such that:
    i. $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$,
    ii. $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ and,
    iii. $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell\leq i}) = 1$.
- *Extractor.* The extractor $\mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})$ proceeds as follows:
  (a) For every $\ell < i$ and $j \in [m_\ell]$ compute $\mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta) := \mathbf{E}_{\mathsf{RS}}(f_{\ell,j})$ and let $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$ be a codeword such that $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ (output $\perp$ if no such codeword exists).
  (b) Compute $\mathbb{w} := \mathbf{E}_{\mathsf{poly}}(\mathbb{y}, \mathbb{x}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell<i}, (\hat{u}_{i,j})_{j\in[m_i]})$ and output $\mathbb{w}$.

  The extractor runs in time at most $O(\mathsf{et}_{\mathsf{poly}} + \sum_{\ell<i} m_\ell \cdot \mathsf{et}_{\mathsf{RS}}) = O(\mathsf{et}_{\mathsf{poly}} + m_{\mathsf{poly}} \cdot \mathsf{et}_{\mathsf{RS}})$.
- *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 $\ell \leq i$, $j \in [m_\ell]$ and pair of distinct codewords $u_{\ell,j}, u'_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$, it holds that $\hat{u}_{\ell,j}(x_\ell) \neq \hat{u}'_{\ell,j}(x_\ell)$. It follows that for every ℓ and j there exists at most one codeword $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_{\ell,j}, \delta)$ 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 $j \in [m_{i\ell}]$ 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 $(u_{\ell,j})_{\ell<i,j\in[m_i]}$ where $u_{\ell,j} \in \mathsf{List}(f_\ell, d_{\ell,j}, \delta)$ and $\hat{u}_{\ell,j}(x_\ell) = y_{i,j}$, it holds that

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell<i}) = 0 \ .$$

Moreover, it holds that $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 1$ only if

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell\leq i}, (\hat{u}_{i,j})_{j\in[m_i]}, \alpha_i) = 1 \ .$$

Thus, we get that

$$\begin{aligned}\Pr_{\alpha_i}\left[\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell<i}, (\hat{u}_{i,j})_{j\in[m_i]}, \alpha_i) = 1\right] &= \Pr_{\alpha_i}\left[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||\alpha_i) = 1\right] \\ &> \varepsilon_i^{\mathsf{piop}} \\ &= \mathsf{err}_i^{\mathsf{poly}}(\mathbb{x}, \mathbb{y}) \ .\end{aligned}$$

It follows by knowledge soundness of the PIOP that

$$((\mathbb{x}, \mathbb{y}), \mathbf{E}(\mathbb{x}, \mathbb{y}, \mathsf{tr})) = ((\mathbb{x}, \mathbb{y}), \mathbf{E}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell\leq i}, (\hat{u}_{i,j})_{j\in[m_i]})) \in \mathcal{R} \ .$$

3. **Bounding $\varepsilon^{\mathsf{com}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := (((f_{i,j})_{j\in[m_i]}, x_i, (y_{i,j})_{j\in[m_\ell]}, \alpha_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}, (A_{i,j}, w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_i]}) \ ,$$

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 $j \in [m_i]$: $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$ 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

$$\Pr_r\left[\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}||r) = 1\right] \leq \varepsilon^{\mathsf{com}} = \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}} m_i \cdot (d - d_i + \mathsf{q}_i + 2)\right) \ .$$

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 $(u_{\ell,j})_{\ell\in[\mathsf{k}_{\mathsf{poly}}],j\in[m_\ell]}$ for which all of the following hold:

(a) $u_{\ell,j} \in \mathsf{List}(f_{\ell,j}, d_\ell, \delta)$,
(b) $\hat{u}_{\ell,j}(x_\ell) = y_{\ell,j}$ and,
(c) $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{\ell,j})_{j\in[m_\ell]}, \alpha_\ell)_{\ell\leq\mathsf{k}_{\mathsf{poly}}}) = 1$.

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 7.4. *There exists $i^* \in [\mathsf{k}_{\mathsf{poly}}]$ and $j^* \in [m_{i^*}]$ such that for every $u_{i^*,j^*} \in \mathsf{List}(f_{i^*,j^*}, d_{i^*,j^*}, \delta)$ 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 $u_{i,j} \in \mathsf{List}(f_{i,j}, d_{i,j}, \delta)$ 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

$$\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{i,j})_{j\in[m_i]}, \alpha_i)_{i\leq \mathsf{k}_{\mathsf{poly}}}) = 1 \ .$$

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: $u_{i,j} \in \mathsf{List}(f_{i,j}, d_{i,j}, \delta)$,
(b) $\hat{u}_{\ell,j}(x_i) = y_{i,j} = \mathsf{Ans}_{i,j}(x_i)$ and,
(c) $\mathsf{State}_{\mathsf{poly}}(\mathbb{x}, \mathbb{y}, ((\hat{u}_{i,j})_{j\in[m_i]}, \alpha_i)_{i\leq \mathsf{k}_{\mathsf{poly}}}) = 1$.

□

Fix i^*, j^* as in Claim 7.4. Together with Lemma 4.4, 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

$$g := \mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[m_i]}\right) \ ,$$

and applying Lemma 4.13, which we can do since $\delta \in (0, \min\{1 - \rho - 1/|\mathcal{L}|, 1 - \mathsf{B}^\star(\rho)\})$:

$$\begin{aligned}
\Pr_{r\leftarrow\mathbb{F}}[\Delta(g, \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \leq \delta] &= \Pr_{r\leftarrow\mathbb{F}}\left[\Delta\left(\mathsf{Combine}\left(d, r, (f'_{i,j}, d_{i,j} - |\mathcal{S}_{i,j}|)_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[m_i]}\right), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]\right) \leq \delta\right] \\
&> \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}}\sum_{j=1}^{m_i}(d - d_{i,j} + |\mathcal{S}_{i,j}| + 1)\right) \\
&\geq \mathsf{err}^\star\left(d, \rho, \delta, \sum_{i=1}^{\mathsf{k}_{\mathsf{poly}}}\sum_{j=1}^{m_i}(d - d_{i,j} + \mathsf{q}_{i,j} + 2)\right) \ .
\end{aligned}$$

Finally,

4. **Bounding $\varepsilon_i^{\mathsf{prx}}$.** At this stage, the partial transcript has the form

$$\mathsf{tr} := (((f_{i,j})_{j\in[m_i]}, x_i, (y_{i,j})_{j\in[m_\ell]}, \alpha_i)_{i\in[\mathsf{k}_{\mathsf{poly}}]}, (A_{i,j}, w_{i,j})_{i\in[\mathsf{k}_{\mathsf{poly}}], j\in[m_i]}, (\pi_\ell, \alpha_{\mathsf{prx},\ell})_{\ell<i}, \pi_i) \ ,$$

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 $j \in [m_i]$: $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$ 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 $i \in [\mathsf{k}_{\mathsf{poly}}]$, $j \in [m_i]$, 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

$$\mathsf{tr} := (((f_{i,j})_{j \in [m_i]}, x_i, (y_{i,j})_{j \in [m_\ell]}, \alpha_i)_{i \in [\mathsf{k}_{\mathsf{poly}}]}, (A_{i,j}, w_{i,j})_{i \in [\mathsf{k}_{\mathsf{poly}}], j \in [m_i]}, (\pi_\ell, \alpha_{\mathsf{prx},\ell})_{\ell < \mathsf{k}_{\mathsf{prx}}},) \ ,$$

   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 $j \in [m_i]$: $f_{i,j}(x) = \mathsf{Ans}_{i,j}(x)$ 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 $\mathsf{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$.

□

A Additional experimental data

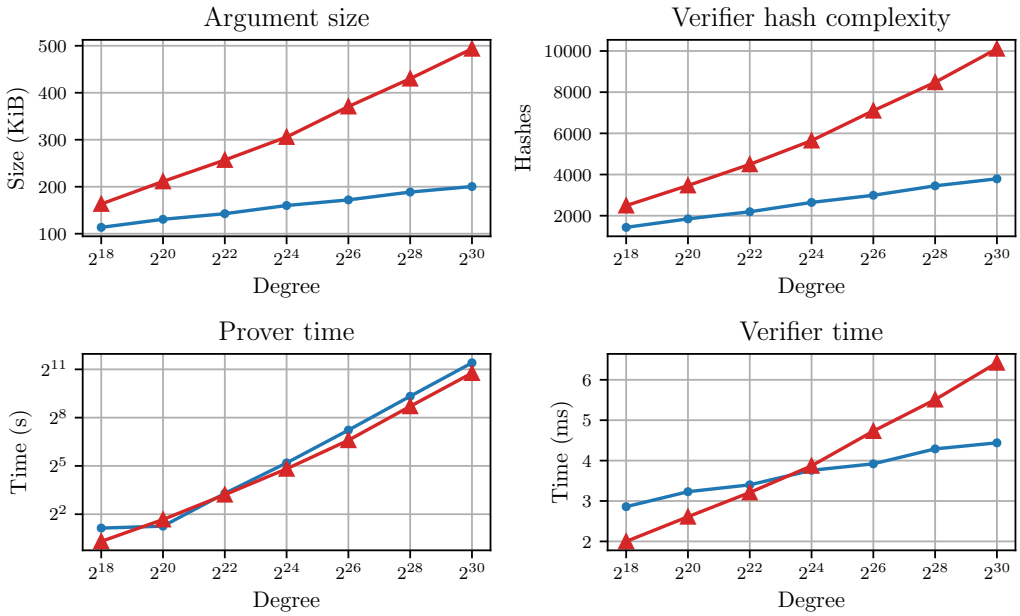
We collected additional experimental data, both in tabular and graphical forms. Figure 3 and Figure 4 further illustrate this information in graphical form, comparing STIR and FRI on each efficiency measure for various rates. Table 4 gives experimental results for the algebraic configuration. Finally, Table 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 ($\frac{\text{FRI}}{\text{STIR}}$)					
$1/2$	$\frac{2562}{1409} \approx 1.82\times$	$\frac{3427}{1842} \approx 1.86\times$	$\frac{4547}{2171} \approx 2.09\times$	$\frac{5718}{2647} \approx 2.16\times$	$\frac{7005}{2971} \approx 2.36\times$	$\frac{8483}{3445} \approx 2.46\times$
$1/4$	$\frac{1680}{1028} \approx 1.63\times$	$\frac{2223}{1318} \approx 1.69\times$	$\frac{2841}{1536} \approx 1.85\times$	$\frac{3445}{1860} \approx 1.85\times$	$\frac{4171}{2039} \approx 2.05\times$	$\frac{5069}{2402} \approx 2.11\times$
$1/8$	$\frac{1369}{841} \approx 1.63\times$	$\frac{1815}{1120} \approx 1.62\times$	$\frac{2232}{1233} \approx 1.81\times$	$\frac{2647}{1554} \approx 1.7\times$	$\frac{3237}{1722} \approx 1.88\times$	$\frac{3877}{1997} \approx 1.94\times$

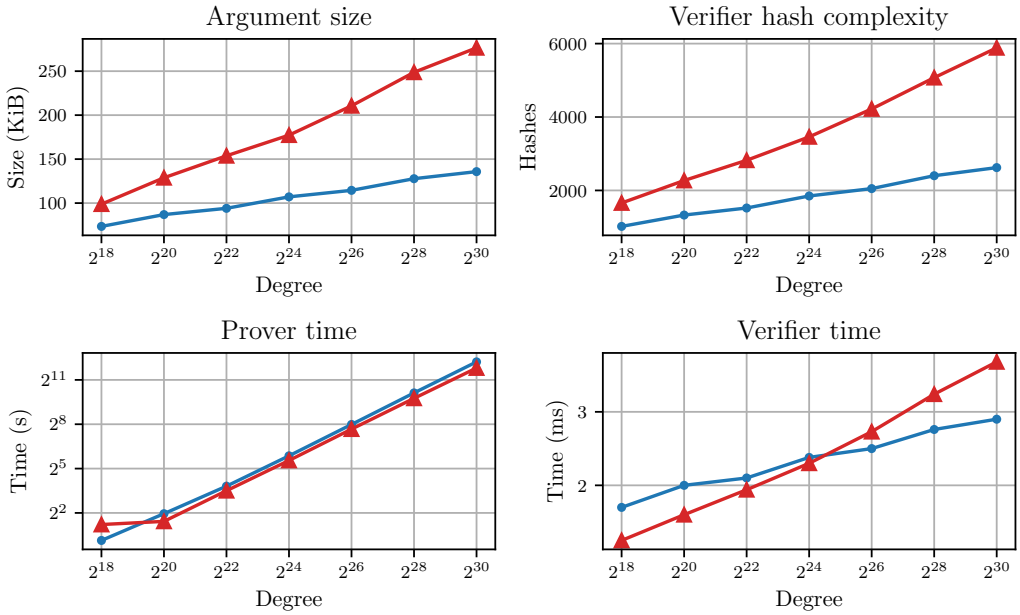
Table 4: Comparison of concrete costs between STIR and FRI when using Poseidon. The numerator of the fraction is the cost associated to FRI, while the denominator is that associated to STIR. For all metrics, lower is better.

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: 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.

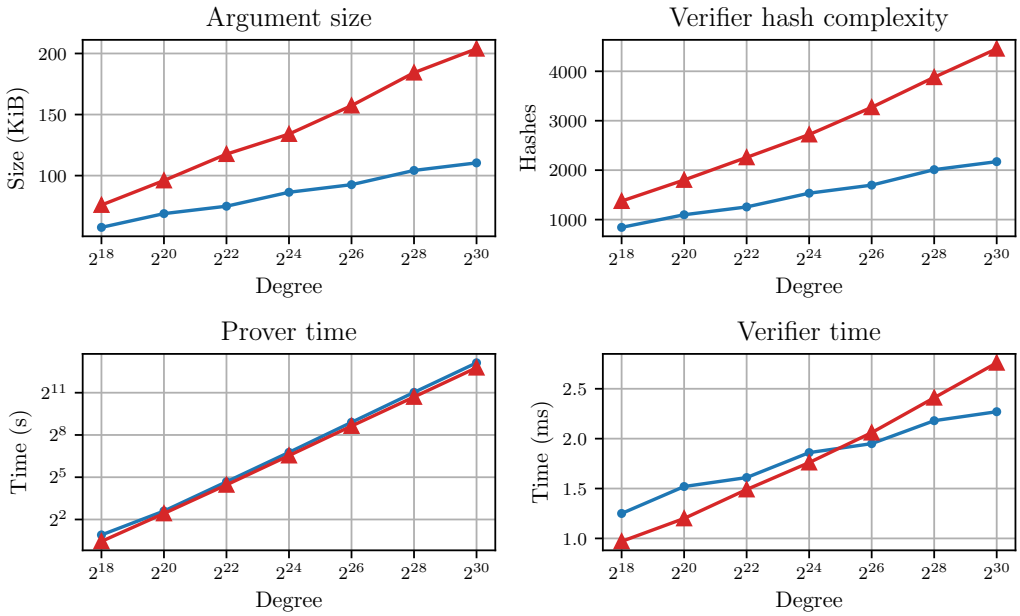


(a) $\rho = 1/2$, FRI: ▲, STIR: ●.

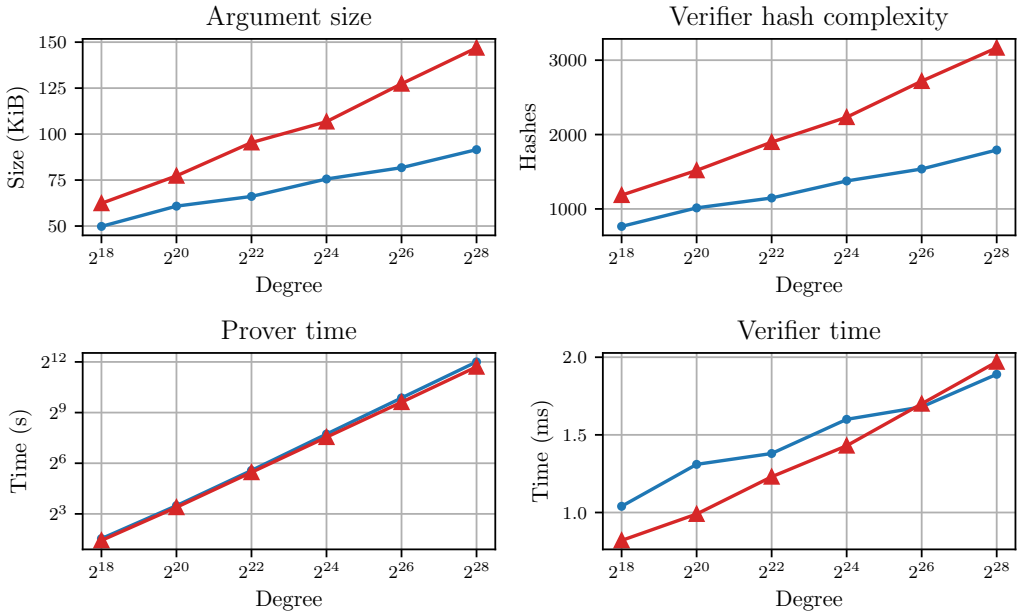


(b) $\rho = 1/4$, FRI: ▲, STIR: ●.

Figure 3: Comparison of FRI and STIR. Figure 3a is for $\rho = 1/2$, Figure 3b for $\rho = 1/4$. Lower is better.



(a) $\rho = 1/8$, FRI: ▲, STIR: ●.



(b) $\rho = 1/16$, FRI: ▲, STIR: ●.

Figure 4: Comparison of FRI and STIR. Figure 4a is for $\rho = 1/8$, Figure 4b for $\rho = 1/16$. Lower is better.

B A poly-IOP for R1CS

We give a polynomial IOP for the R1CS relation.

Definition B.1. *The relation $\mathcal{R}_{\mathsf{R1CS}}$ is the set of all pairs $((\mathbb{F}, k, n, m, A, B, C, v), w)$ where $\mathbb{F}$ is a finite field, $k, n, m \in \mathbb{N}$ denote the number of inputs, variables and constraints respectively (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 B.2. *There is an IOP for $\mathcal{R}_{\mathsf{R1CS}}$ with the following properties.*
- Round complexity: *2.*
- Number of polynomials: 6 *with degree at most n 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 $\varepsilon^{\mathsf{dec}} \leq \frac{2n}{|\mathbb{F}|}$.*

B.1 Construction

Construction B.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 $\{1, \ldots, k\}$.
- $\hat{V}_H \in \mathbb{F}^{<n}[X]$ and $\hat{V}_{H_{\mathsf{in}}} \in \mathbb{F}^{<k}[X]$ 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 polynomial such that $\hat{q}_{M,r}(x) := r^x \cdot \sum_{a \in H} M^\top(a, x)$ for every $x \in H$.

The IOP proceeds as follows.

- **Inputs:** The honest prover is given $((\mathbb{F}, k, n, m, A, B, C, v), w) \in \mathcal{R}_{\mathsf{R1CS}}$, and the verifier is given $(\mathbb{F}, k, n, m, A, B, C, v)$.

- **Interaction phase:**

  1. **Commit to witness polynomials:** The prover sends polynomials $\hat{f}_A, \hat{f}_B, \hat{f}_C, \hat{f}_0, \hat{f}_w \in \mathbb{F}^{<n}[X]$. 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) $\hat{f}_w(X) := \frac{\hat{w}(X) - \hat{v}(X)}{\hat{V}_{H_{\mathsf{in}}}(X)}$ where $\hat{v} \in \mathbb{F}^{<n}[X]$ is the unique low-degree polynomial that is equal to v on H_{in} and 0 on H_{aux}.

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)

$$\begin{aligned}\hat{u}(X) := \; & \hat{p}_r(X) \cdot \hat{f}_A(X) - \hat{q}_{A,r}(X) \cdot \hat{f}_z(X) \\ & + r^n \cdot \Big(\hat{p}_r(X) \cdot \hat{f}_B(X) - \hat{q}_{B,r}(X) \cdot \hat{f}_z(X)\Big) \\ & + r^{2n} \cdot \Big(\hat{p}_r(X) \cdot \hat{f}_C(X) - \hat{q}_{C,r}(X) \cdot \hat{f}_z(X)\Big)\end{aligned}$$

   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 $a \leftarrow \mathbb{F}$ 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 a. Accept if the following checks pass.
  1. **Zero test:** $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{f}_0(a) \cdot \hat{V}_{H_{\mathsf{in}}}(a)$.
  2. **Univariate sumcheck test:** $\hat{u}(a) = \hat{g}_1(a) \cdot \hat{V}_H(a) + a \cdot \hat{g}_2(a)$. Evaluating $\hat{u}(a)$ on reduces to computing:
     (a) $\hat{f}_z(a) := \hat{f}_w(a) \cdot \hat{V}_H(a) + \hat{v}(a)$.
     (b)

$$\begin{aligned}\hat{u}(a) := \; & \hat{p}_r(a) \cdot \hat{f}_A(a) - \hat{q}_{A,r}(a) \cdot \hat{f}_z(a) \\ & + r^n \cdot \Big(\hat{p}_r(a) \cdot \hat{f}_B(a) - \hat{q}_{B,r}(a) \cdot \hat{f}_z(a)\Big) \\ & + r^{2n} \cdot \Big(\hat{p}_r(a) \cdot \hat{f}_C(a) - \hat{q}_{C,r}(a) \cdot \hat{f}_z(a)\Big)\end{aligned}$$

     (The verifier can compute $\hat{V}_H(a)$, $\hat{v}(a)$, $\hat{p}_r(a)$, $\hat{q}_{A,r}(a)$, $\hat{q}_{B,r}(a)$, and $\hat{q}_{C,r}(a)$ 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.

B.2 Round-by-round knowledge soundness

Lemma B.4. *The poly-IOP in Construction B.3 has round-by-round knowledge soundness errors* $(\varepsilon^{\mathsf{shift}}, \varepsilon^{\mathsf{dec}})$ *with extraction time* $O(n \cdot (n-k))$ *where:*

- $\varepsilon^{\mathsf{shift}} \leq \frac{3n}{|\mathbb{F}|}$.
- $\varepsilon^{\mathsf{dec}} \leq \frac{2n}{|\mathbb{F}|}$.

Preliminaries. In the proof we use the following fact, showing another description of one of the polynomials described in the protocol:

Fact B.5. *For every $r \in \mathbb{F}$:*

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

Proof. The fact follows by opening up the expressions:

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

□

We use the following lemma, implementing a univariate sumcheck, first shown in [BCRSVW19], and described in the form below in [ACY23]:

Lemma B.6 ([BCRSVW19])**.** *Let H be a multiplicative subgroup of $\mathbb{F}^*$. Let $\hat{f} \in \mathbb{F}^{<d}[X]$ be a polynomial, and $s \in \mathbb{F}$ be a claimed sum. Then:*

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

$$\Pr_{a \leftarrow \mathbb{F}} \left[\begin{array}{l} \hat{g}_1 \in \mathbb{F}^{<|H|-1}[X] \\ \wedge\ \hat{g}_2 \in \mathbb{F}^{<d-|H|+1}[X] \\ \wedge\ \hat{f}(a) = \hat{g}_1(a) \cdot \hat{V}_H(a) + (a \cdot \hat{g}_2(a) + s/|H|) \end{array} \middle| \ \hat{f}(X) \equiv \hat{g}_1(X) \cdot \hat{V}_H(X) + (X \cdot \hat{g}_2(X) + s/|H|) \right] = 1 \ .$$

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

$$\Pr_{a \leftarrow \mathbb{F}} \left[\begin{array}{l} \hat{g}_1 \in \mathbb{F}^{<|H|-1}[X] \\ \wedge\ \hat{g}_2 \in \mathbb{F}^{<d-|H|+1}[X] \\ \wedge\ \hat{f}(a) = \hat{g}_1(a) \cdot \hat{V}_H(a) + (a \cdot \hat{g}_2(a) + s/|H|) \end{array} \middle| \ (\hat{g}_1, \hat{g}_2) \leftarrow \tilde{\mathbf{P}} \right] \leq \frac{d}{|\mathbb{F}|} \ .$$

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).

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 $\mathbb{x} := (\mathbb{F}, k, n, m, A, B, C, v)$ 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 $\hat{w}(X) := \hat{f}_w(X) \cdot \hat{V}_{H_{\mathsf{in}}}(X) + \hat{v}(X)$ and then outputs $w \colon H \setminus H_{\mathsf{in}} \to \mathbb{F}$, where $w(i) = \hat{w}(i)$. 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:

$$\begin{aligned}\hat{h}(X) := \ & \sum_{\alpha \in H} \left(\hat{f}_A(\alpha) - \sum_{\beta \in H} A(\alpha, \beta) \cdot \hat{f}_z(\beta) \right) \cdot X^{\alpha} \\ & + \sum_{\alpha \in H} \left(\hat{f}_B(\alpha) - \sum_{\beta \in H} B(\alpha, \beta) \cdot \hat{f}_z(\beta) \right) \cdot X^{n+\alpha} \\ & + \sum_{\alpha \in H} \left(\hat{f}_C(\alpha) - \sum_{\beta \in H} C(\alpha, \beta) \cdot \hat{f}_z(\beta) \right) \cdot X^{2n+\alpha} \ .\end{aligned}$$

     Observe that $\mathsf{deg}(\hat{h}) \leq 2n + |H| = 3n$, and that for r chosen by the verifier, by Fact B.5 the evaluation $\hat{h}(r)$ is equivalent to $\sum_{\alpha \in H} \hat{u}(\alpha)$. 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 $M \in \{A, B, C\}$:

$$\hat{f}_M(X) = \sum_{\beta \in H} M(X, \beta) \cdot \hat{f}_z(\beta) = \sum_{\beta \in H} M(X, \beta) \cdot \left(\hat{f}_w(\beta) \cdot \hat{V}_{H_{\mathsf{in}}}(\beta) + \hat{v}(\beta) \right) \ .$$

     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 $\alpha \in H$ it holds that

$$\hat{f}_M(\alpha) = \sum_{\beta \in H} M(\alpha, \beta) \cdot z(\beta) \ .$$

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 $\alpha \in H$, $\hat{f}_A(\alpha) \cdot \hat{f}_B(\alpha) = \hat{f}_C(\alpha)$. Therefore, for every $\alpha \in H$,

$$\left(\sum_{\beta \in H} A(\alpha, \beta) \cdot z(\beta)\right) \cdot \left(\sum_{\beta \in H} B(\alpha, \beta) \cdot z(\beta)\right) = \hat{f}_A(\alpha) \cdot \hat{f}_B(\alpha) = \hat{f}_C(\alpha) = \left(\sum_{\beta \in H} C(\alpha, \beta) \cdot z(\beta)\right) \quad .$$

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 a.

   - *State function.* We set $\mathsf{State}(\mathbb{x}, \mathsf{tr}||a) = 1$ if and only if both of the following hold:
     (a) $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{f}_0(a) \cdot \hat{V}_H(a)$ and
     (b) $\hat{u}(a) = \hat{g}_1(a) \cdot \hat{V}_H(a) + a \cdot \hat{g}_2(a)$, 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 $\mathsf{State}(\mathbb{x}, \mathsf{tr}||a) = 0$, 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

$$\Pr_r \left[\mathsf{State}(\mathbb{x}, \mathsf{tr}||r) = 1\right] \leq \varepsilon^{\mathsf{dec}} = \frac{2n}{|\mathbb{F}|} \quad .$$

     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 $a \leftarrow \mathbb{F}$ it holds that $\hat{f}_A(a) \cdot \hat{f}_B(a) - \hat{f}_C(a) = \hat{f}_0(a) \cdot \hat{V}_H(a)$ 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, $\mathsf{State}(\mathbb{x}, \mathsf{tr}||a) = 1$ with probability at most $2n/|\mathbb{F}|$.
     – If $\sum_{a \in H} \hat{u}(a) \neq 0$, then by Lemma B.6, for every $\hat{g}_1 \in \mathbb{F}^{<n}[X]$ and $\hat{g}_2 \in \mathbb{F}^{<n-1}[X]$, the probability that $\hat{u}(a) = \hat{g}_1(a) \cdot \hat{V}_H(a) + a \cdot \hat{g}_2(a)$ is at most $2n/|\mathbb{F}|$. Thus in this case, by Item 2b, $\mathsf{State}(\mathbb{x}, \mathsf{tr}||a) = 1$ 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 $2n/|\mathbb{F}|$.

C Derivations for Section 5.3

We derive bounds on the parameters of the IOPPs described in Section 5.3.

- In Appendix C.1 we give derivations for computing provable security bounds.
- In Appendix C.2 we give derivations for computing security bounds assuming Conjecture 5.6 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 C.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 C.2. $\rho_i \geq \rho/d$.

Proof. $\rho_i = (2/k)^i \cdot \rho \geq \rho/k^M = \rho/k^{\lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor} \geq \rho/d$. □

C.1 Provable security

We bound the properties of the IOPP for provable soundness error described in Section 5.3. 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 Remark 5.3.

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 Fact C.1.

Round-by-round soundness. We begin by confirming the requirements needed in order to apply Lemma 5.4 using $\mathsf{B}^\star(\rho) = \sqrt{\rho}$ as defined in Theorem 4.1.

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1-\sqrt{\rho_0})$: 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 3.4), 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

$$\mathsf{err}^\star(d, \rho, \delta, m) := \frac{(m-1)\cdot d^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{1-\sqrt{\rho}-\delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7} \ ,$$

as in Theorem 4.1:

- $\varepsilon^{\mathsf{fold}}$:

$$\begin{aligned}
\varepsilon^{\mathsf{fold}} &\leq \mathsf{err}^{\star}(d_0/k_0, \rho_0, \delta_0, k_0) \\
&= \frac{(k-1)\cdot(d/k)^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{1-\sqrt{\rho}-\delta', \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1)\cdot(d/k)^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{\max\left\{1-\sqrt{\rho}-\delta, \eta_0\right\}, \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1)\cdot(d/k)^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{\eta_0, \sqrt{\rho}/20\right\}\right)^7} \leq 2^{-\lambda} \\
&\leq \frac{(k-1)\cdot(d/k)^2}{|\mathbb{F}|\cdot(2\cdot\eta_0)^7} \\
&\leq 2^{-\lambda} \ ,
\end{aligned}$$

  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 $\varepsilon^{\mathsf{fin}} \leq (1-\delta_M)^{t_M} = (\sqrt{\rho_M}+\eta_M)^{\left\lceil\frac{\lambda}{-\log(\sqrt{\rho_M}+\eta_M)}\right\rceil} \leq 2^{-\lambda}$.

C.2 Conjectured security

We bound the properties of the IOPP described in Section 5.3 when assuming Conjecture 5.6. 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 Remark 5.3.

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.

- $i=0$:

$$\eta_0 = \left(\frac{2^\lambda\cdot(k-1)^{c_2}\cdot(d/k)^{c_2}}{\rho^{c_1+c_2}\cdot|\mathbb{F}|}\right)^{1/c_1} = \frac{2^\lambda\cdot(k-1)\cdot(d/k)}{\rho^2\cdot|\mathbb{F}|} < \frac{2^\lambda\cdot d}{\rho^2}\cdot\frac{1}{2^{\lambda+1}\cdot|\mathcal{L}|^3} < \frac{\rho}{2} \ .$$

- $i>0$:

$$\begin{aligned}\eta_i &:= \max\left\{\frac{2\cdot\rho_i}{d_i},\frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s}\right)^{\frac{1}{2\cdot c_3}},\left(\frac{2^{\lambda+1}\cdot d_i^{c_2}}{\rho_i^{c_1+c_2}\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+s-1)^{c_2}+\left(\frac{k-1}{k}\right)^{c_2}\right)\right)^{1/c_1}\right\}\\ &= \max\left\{\frac{2\cdot\rho_i}{d_i},\frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^2}\right)^{\frac{1}{2}},\frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+1)+\left(\frac{k-1}{k}\right)\right)\right\} \ .\end{aligned}$$

  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 $4\leq d_{\mathsf{stop}}\leq d_i$.
  - Second option:

$$\frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^2}\right)^{\frac{1}{2}} < |\mathcal{L}|\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}|)^2}\right)^{\frac{1}{2}} \leq |\mathcal{L}|\cdot\left(\frac{2^\lambda\cdot d^2}{4\cdot 2^\lambda\cdot d^2\cdot|\mathcal{L}|^6}\right)^{\frac{1}{2}} < \frac{\rho}{2\cdot d} < \frac{\rho_i}{2} \ .$$

  - 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.* $2M + 1 = 2 \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + 1$.
- *Proof length.* $M \cdot s + \dfrac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} |\mathcal{L}_i|$

$$\begin{aligned} &= 2 \cdot s \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + \frac{d}{k^{\lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor}} + \sum_{i=1}^{\lfloor \log_k d \rfloor} \frac{|\mathcal{L}|}{2^i} \\ &\leq |\mathcal{L}| + 2 \cdot s \cdot \lfloor \log_k(d/d_{\mathsf{stop}}) \rfloor + k \cdot d_{\mathsf{stop}} - 1 \ . \end{aligned}$$

- *Input query complexity.* $t_0 \leq \frac{\lambda}{-\log(1-\delta')}$.
- *Proof query complexity.* Observe that $\frac{\log(1.5 \cdot \rho)}{\log(k/2)} < 0$ 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 Fact C.1.

Round-by-round soundness. We begin by confirming the requirements needed in order to apply Lemma 5.4 using $\mathsf{B}^\star(\rho) = \rho$ as defined in Conjecture 5.6.

- $\delta_0 \in (0, \Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1 - \rho_0)$: 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 Conjecture 5.6, 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

$$\mathsf{err}^\star(d, \rho, \delta, m) := \frac{(m-1)^{c_2} \cdot d^{c_2}}{\eta^{c_1} \cdot \rho^{c_1+c_2} \cdot |\mathbb{F}|} \ ,$$

as in Conjecture 5.6:

- $\varepsilon^{\mathsf{fold}}$:

$$\begin{aligned}\varepsilon^{\mathsf{fold}} &\leq \mathsf{err}^\star(d_0/k_0, \rho_0, \delta_0, k_0) \\ &= \frac{(k-1)^{c_2} \cdot (d/k)^{c_2}}{\eta_0^{c_1} \cdot \rho^{c_1+c_2} \cdot |\mathbb{F}|} \\ &\leq 2^{-\lambda} \ ,\end{aligned}$$

  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 $\varepsilon^{\mathsf{fin}} \leq (1-\delta_M)^{t_M} = (\rho_M + \eta_M)^{\left\lceil \frac{\lambda}{-\log(\rho_M+\eta_M)} \right\rceil} \leq 2^{-\lambda}$.

Acknowledgments

We thank Marcin Górny, Francesco Intoci, Pratyush Mishra and Andrew Zitek-Estrada for assisting with the Merkle tree and Fast Fourier Transform implementations in `arkworks`.

Gal Arnon is supported in part by a grant from the Israel Science Foundation (Grant No. 2686/20), by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness, and by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center. Alessandro Chiesa and Giacomo Fenzi are supported in part by the Ethereum Foundation. Eylon Yogev is supported by an Alon Young Faculty Fellowship, by the Israel Science Foundation (Grant No. 2302/22).

References

[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.

[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.

[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.

[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.

[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.

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

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

[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.

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

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

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

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

[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.

[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.

[Her] *Hermez*. `https://hermez.io`.

[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.

[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.

[Nep] *Netpture*. `https://neptune.cash/`.

[Ola] *Ola*. `https://ola.finance`. Accessed: insert date here.

[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.

[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.

[Ris] *Risc0*. https://risc0.com.

[San] *Sandstorm*. https://github.com/andrewmilson/sandstorm.

[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.

[Zks] *zkSync*. https://zksync.io.

[ark] arkworks. *An ecosystem for developing and programming with zkSNARKs*. arkworks.rs.