STIR: Reed–Solomon Proximity Testing with Fewer Queries

GAL ARNON, Weizmann Institute of Science Faculty of Mathematics and Computer Science, Israel
ALESSANDRO CHIESA, EPFL, Switzerland
GIACOMO FENZI, EPFL, Switzerland
EYLON YOGEV, Bar-Ilan University Faculty of Exact Sciences, Israel

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 $O(\log d + \lambda \cdot \operatorname{loglog} d)$, 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.

CCS Concepts: • **Theory of computation → Computational complexity and cryptography**; **Cryptographic protocols**; **Interactive proof systems**.

Additional Key Words and Phrases: Interactive oracle proofs, Reed–Solomon proximity testing

ACM Reference Format:
Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. 2026. STIR: Reed–Solomon Proximity Testing with Fewer Queries. *J. ACM* 73, 6, Article 42 (December 2026), 38 pages. https://doi.org/10.1145/3858414

1 Introduction

Reed–Solomon (**RS**) codes [34] 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 $f : \mathcal{L} \to \mathbb{F}$ 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 $f : \mathcal{L} \to \mathbb{F}$ 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.

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).

Authors' Contact Information: Gal Arnon, Weizmann Institute of Science Faculty of Mathematics and Computer Science, Rehovot, Israel; e-mail: galarnon42@gmail.com; Alessandro Chiesa, EPFL, Lausanne, Switzerland; e-mail: alessandro.chiesa@epfl.ch; Giacomo Fenzi (corresponding author), EPFL, Lausanne, Switzerland; e-mail: giacomo.fenzi@epfl.ch; Eylon Yogev, Computer Science, Bar-Ilan University Faculty of Exact Sciences, Ramat Gan, Israel; e-mail: eylon.yogev@biu.ac.il.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-735X/2026/12-ART42
https://doi.org/10.1145/3858414

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 [24, 27, 33] and more. Moreover, in practice, it enables highly efficient constructions of **succinct non-interactive arguments (SNARGs)** [19, 20, 23].

Especially noteworthy is FRI (and its variants), which is an IOPP for RS codes that enjoys practical efficiency [19, 20, 23]. The most practically efficient version of FRI is given in [20] and its implementation underlies numerous SNARG-based real-world systems, including [1–10]. 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 [22], 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 that 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 [25, 38].

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* $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ *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* $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ *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 the sense that the rate shrinks (with each recursion), which improves soundness and reduces the query complexity. In particular, a round of STIR reduces testing proximity to $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing

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

Table 1. A Comparison of FRI and STIR

	Queries	Verifier Time
FRI	$q_{\mathsf{FRI}} \coloneqq O\left(\lambda + \frac{\lambda}{k} \cdot \log d\right)$	$O\left(q_{\mathsf{FRI}} \cdot 2^k\right)$
STIR	$q_{\mathsf{STIR}} \coloneqq O\left(\lambda + \frac{\lambda}{k} \cdot \log \frac{\log d}{k}\right)$	$O\left(q_{\mathsf{STIR}} \cdot 2^k + \frac{\lambda^2}{k} \cdot 2^k\right)$

Here, λ is the security parameter, k is the folding parameter, and d is the number degree of the polynomial. The dependence on the rate ρ is suppressed, and we assume that $\log d \leq \lambda$. Running time is measured in field operations.

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 [19] is a concretely efficient IOPP for RS codes. A variant called DEEP-FRI [23] achieves better soundness while essentially preserving other efficiency measures. Later, [20] improved the analysis of the [19] 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 and STIR have similar structures: they both have $O(\log_k d)$ 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. An overview of how STIR works differently from FRI is given in Section 2.1.

However, these advantages entail certain tradeoffs relative to FRI. While the STIR prover exhibits a slight increase in running time, this overhead is typically negligible in practice, as the dominant cost remains the commitment to the function. The STIR verifier, however, is algebraically more complex: although it performs fewer hash evaluations than the FRI verifier, it requires more field operations, which may increase the total verification time in certain parameter regimes. In particular, let q_{STIR} and q_{FRI} denote the query complexities of STIR and FRI, respectively. The STIR verifier performs $O\left(q_{\mathsf{STIR}} \cdot 2^k + \frac{\lambda^2}{k} \cdot 2^k\right)$ field operations, whereas the FRI verifier requires $O\left(q_{\mathsf{FRI}} \cdot 2^k\right)$ field operations. We refer to Table 1 for a detailed comparison.

The work of [14] 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) [14] 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 [14] can be made concretely efficient.

The core concepts underlying STIR have also proven valuable in the multivariate setting. Notably, the WHIR protocol [13] adapts techniques from STIR and integrates them into the BaseFold [39] multivariate IOP proximity test (alongside other optimizations) to achieve a highly efficient multivariate proximity tester. Proximity testing has also been investigated for other code families, such as in [26, 30].

Table 2. Comparison of STIR and FRI

	$d = 2^{22}, \rho = 1/4$		$d = 2^{28}, \rho = 1/2$	
	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

Experimental results. We implement STIR in Rust using the `arkworks` [11] 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×.

Our experiments show that the running time of the prover is slightly larger than that of FRI. However, typically, proximity tests are used in a batched setting, where the prover's running time is primarily constrained by computing the initial commitment (prior to running the proximity test). In this setting, replacing FRI with STIR leaves the prover time essentially unchanged.

We give concrete examples in Table 2. 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.

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* [2] for the language of interest (or other related IOP variants) and an RS proximity test (such as FRI or STIR or some other protocol). Prior compilers [14, 21] 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 how to use our new compiler, we also give a polynomial IOP for the NP-complete language R1CS similar to the one given in [21] 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

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

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.

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 $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$ and degrees $d_1, \ldots, d_m, d^*$ with $d^* \geq \max_{i \in [m]}\{d_i\}$, define a function $f^* : \mathcal{L} \to \mathbb{F}$ (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. [20] 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. [21] 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, [14] 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 [21]).

Theorem 2 (informal). *There is a probabilistic transformation* Combine *such that for every functions $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$, 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*

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

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*

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

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 [20, 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" $k \geq 4$ (a power of 2) and a query repetition parameter t. An iteration of STIR reduces testing proximity to the code $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ to testing proximity to a related code $\mathcal{C}' := \mathsf{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $|\mathcal{L}'| = n/2$, i.e., STIR reduces the degree by a factor of k while decreasing the size of the evaluation domain only by a factor of 2.

Testing proximity to $\mathcal{C}'$ is easier than testing proximity to $\mathcal{C}$ for two reasons: (i) $\mathcal{C}'$ is "smaller" than $\mathcal{C}$ in the sense that the degree is reduced from d to d/k and the evaluation domain size is reduced from n to $n/2$; (ii) the rate of the code is reduced from $\rho = \frac{d}{|\mathcal{L}|}$ to $\rho' = \frac{d/k}{|\mathcal{L}'|} = \frac{2}{k} \cdot \rho$. The first reason, which is enjoyed by prior IOPPs such as FRI, iteratively simplifies the code being tested until a trivial test suffices. The second reason is novel to STIR. Intuitively, testing proximity to a code with a smaller rate is easier because the code has more "redundancy". Indeed, rate reduction in STIR is the key feature that facilitates smaller query complexity (the smaller the rate, the fewer queries are needed to achieve a target soundness error), as we discuss later in this section. Similar ideas of rate reduction have proven useful elsewhere to achieve efficient IOPPs [36, 37].

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 $\mathcal{C}$ (for any $\delta \in (0, 1 - \sqrt{\rho})$), 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 $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$.

The query complexity of STIR. 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

[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.

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 $t_0 := \frac{\lambda}{-\log(1-\delta)}$ (recall that $\delta \in (0, 1-\sqrt{\rho})$) and $t_i := \frac{\lambda}{-\log\sqrt{\rho_i}}$ (ignoring rounding issues) to get error $2^{-\lambda}$. Then, the query complexity in each iteration decreases: since $\mathsf{RS}[\mathbb{F}, \mathcal{L}_i, d/k^i]$ has rate

$$\rho_i := \frac{d}{k^i} \cdot \frac{1}{|\mathcal{L}_i|} = \left(\frac{2}{k}\right)^i \cdot \frac{d}{n} = \left(\frac{2}{k}\right)^i \cdot \rho \ .$$

the query complexity at round i is

$$t_i := \frac{\lambda}{-\log\left((2/k)^{i/2} \cdot \sqrt{\rho}\right)} = \frac{2 \cdot \lambda}{i \cdot \log(k/2) - 2 \cdot \log\sqrt{\rho}} \ .$$

Note that t_i decreases as i increases. 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 [19] 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 $f : \langle\omega\rangle \to \mathbb{F}$ at a point $r \in \mathbb{F}$ is a function $f_r := \mathsf{Fold}(f, r) : \langle\omega^k\rangle \to \mathbb{F}$. 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., [14, 19, 23]), and has the following properties (see Section 4.4 for a proof of these properties):

(1) If $f \in \mathsf{RS}[\mathbb{F}, \langle\omega\rangle, d]$, then, for every r, $f_r \in \mathsf{RS}[\mathbb{F}, \langle\omega^k\rangle, d/k]$; and
(2) If f is δ-far from $\mathsf{RS}[\mathbb{F}, \langle\omega\rangle, d]$ for $\delta \in (0, 1 - \sqrt{\rho})$, then with probability at least $1 - \mathrm{poly}(|\mathcal{L}|)/|\mathbb{F}|$ over the choice of r, f_r is δ-far from $\mathsf{RS}[\mathbb{F}, \langle\omega^k\rangle, d/k]$.

Furthermore, this mapping is local: given r and oracle access to f, $\mathsf{Fold}(f, r)$ can be computed at any point by reading a k tuple of field elements from f.

Univariate function quotienting. We define a notion of quotienting for univariate functions, inspired by the "DEEP" technique of [23]. The *quotient* of a function $f : \langle\omega\rangle \to \mathbb{F}$ relative to $p : S \to \mathbb{F}$ 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 $g : \mathcal{L}' \to \mathbb{F}$. 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.
(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 $p : \mathcal{G} \to \mathbb{F}$ 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.

[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.

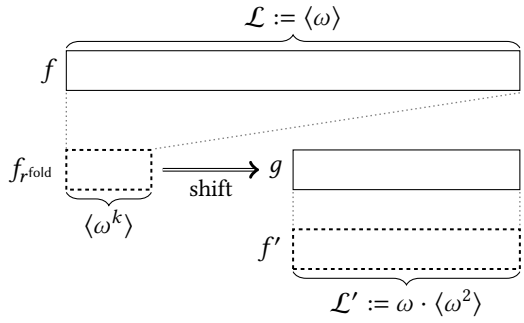


Fig. 1. The basic structure of STIR: given r^{fold}, the function f virtually defines $f_{r^{\text{fold}}} := \text{Fold}(f, r^{\text{fold}})$, which is (virtually) evaluated over $\langle\omega^k\rangle$. The prover then sends g evaluated over the larger domain $\mathcal{L}'$. Finally, the function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

The protocol has perfect completeness, which directly follows from the honest prover's strategy described above and the properties of the fold and quotient operations. We discuss the complexity measures 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 (informal). *If f is δ-far from $\text{RS}[\mathbb{F}, \mathcal{L}, d]$ (for $\delta \in (0, 1-\sqrt{\rho})$) then f' is (approximately) $(1-\sqrt{\rho'})$-far from the code $\text{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$, except with probability $(1-\delta)^t + \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$.*

Proof (Proof sketch).

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

(2) With high probability there is at most one codeword at distance (roughly) $1-\sqrt{\rho'}$ from g that evaluates to β at r^{out}. In more detail, with probability $1 - \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$ there exists at most one codeword $u \in \text{RS}[\mathbb{F}, \mathcal{L}', d/k]$ where $\Delta(g, u)$ is bounded by $1-\sqrt{\rho'}$ (up to small slack parameters) and, additionally, $\hat{u}(r^{\text{out}}) = \beta$, where $\hat{u}$ is the extension of u to a unique degree d/k polynomial.
To see this, by the Johnson bound, the code $\text{RS}[\mathbb{F}, \mathcal{L}', d/k]$ is (γ, ℓ)-list-decodable for $\gamma \approx 1-\sqrt{\rho'}$ and $\ell = \text{poly}(|\mathcal{L}'|) = \text{poly}(|\mathcal{L}|)$, meaning that there are at most ℓ polynomials of degree less than d/k at distance γ to g. Each pair of such polynomials agree on less than d/k points, and so the total number of points in $\mathbb{F}$ for which there exist two distinct polynomials that are γ-close to g that agree on these points is bounded by $\binom{\ell}{2} \cdot d/k = O(\ell^2 \cdot d/k)$. One such point is sampled from $\mathbb{F}$ with probability at most $O(\ell^2 \cdot d/(k \cdot |\mathbb{F}|)) = \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$ (recall that $d < |\mathcal{L}|$).

If Items 2.2 and 2.2 both hold, which happens with probability $1 - \text{poly}(|\mathcal{L}|)/|\mathbb{F}|$, then f' is (approximately) $(1-\sqrt{\rho'})$-far from the code $\text{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$ 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 2.2, then $f' := \text{Quotient}(g, \mathcal{G}, p)$ is $(1-\sqrt{\rho'})$-far from $\text{RS}[\mathbb{F}, \mathcal{L}', d/k - |\mathcal{G}|]$ since $p(r^{\text{out}}) = \beta$.

— If there exists a codeword u as in Item 2.2, then by Item 2.2 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 $f_{r_i^{\mathsf{fold}}}(r_i^{\mathsf{shift}}) \neq \hat{u}(r_i^{\mathsf{shift}})$, 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 $\max\left\{\frac{\mathrm{poly}(|\mathcal{L}|)}{|\mathbb{F}|}, (1-\delta)^t\right\}$. 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 $f : \mathcal{L} \to \mathbb{F}$, 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.

Prior solutions. Degree correction was tackled (implicitly) in [21] and in [14]. 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 $\rho := d^*/|\mathcal{L}|$, [21] show that this works provided that $\delta < \frac{1-2\cdot\rho}{2}$, and [14] 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 2.3 holds by construction. Next, we sketch the proof that Item 2.3 holds provided that $\delta < \min\{1-\sqrt{\rho}, 1-(1+1/d^*)\cdot\rho\}$.

[20] show that if with probability at least $\mathrm{err}^*(d^*, \rho, \delta, e+1)$ (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 $\widehat{f}_i$ of degree less than d^* such that $f_i(x) = \widehat{f}_i(x)$ for every $x \in S$. See [20], Theorem 1.5 and this article at Section 4.1 for further details.

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

Proof (Proof sketch). We prove the claim via (reverse) induction on i. The base case is immediate since $\deg(\widehat{f}_e) < d^*$. Assuming that $\deg(\widehat{f}_{i+1}) < d^* - e + i + 1$, we show that $\deg(\widehat{f}_i) < d^* - e + i$. Consider the polynomial $\hat{p}(X) := X \cdot \widehat{f}_i(X)$ and observe that, since $\deg(\widehat{f}_i) < d^*$, it holds that $\deg(\hat{p}) < d^* + 1$. By correlated agreement on S, for every $x \in S$:

$$\hat{p}(x) = x \cdot \widehat{f}_i(x) = x \cdot f_i(x) = x^{i+1} \cdot f(x) = f_{i+1}(x) = \widehat{f}_{i+1}(x) \ .$$

We conclude that $\hat{p}$ and $\widehat{f}_{i+1}$ agree on all points of S. Since $|S| \geq (1-\delta)\cdot|\mathcal{L}| > (1+1/d^*)\cdot\rho\cdot|\mathcal{L}| = d^*+1$ and $\deg(\hat{p}), \deg(\widehat{f}_{i+1}) < d^* + 1$, it follows that $\hat{p}$ and $\widehat{f}_{i+1}$ are identical. In particular, $\deg(\hat{p}) = \deg(\widehat{f}_{i+1}) < d^* - e + i + 1$. Recalling that $\hat{p}(X) := X \cdot \widehat{f}_i(X)$, it follows that $\deg(\widehat{f}_i) < d^* - e + i$.

Efficient evaluation of f^*. At first glance, the technique described above does not allow for efficient local access to f^* as described in Item 2.3. Indeed, as the sum of $e+1$ different functions, it naively takes $O(e)$ field operations 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 $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$ 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 $O(\sum_i \log e_i) = O(m \cdot \log d^*)$ operations given local access to $f_1, \ldots, f_m$.

3 Preliminaries

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

- The "hat" symbol over a function (e.g., $\hat{p}$) denotes that it is a polynomial.
- For two functions $f, g : \mathcal{L} \to \mathbb{F}$, $\Delta(f, g)$ is the fractional Hamming distance between f and g (the fraction of points on 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)** [22, 35] are information-theoretic proof systems that combine aspects of Interactive Proofs [17, 29] and Probabilistically Checkable Proofs [15, 16, 18, 28], and also generalize the notion of Interactive PCPs [32]. 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 article, 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 ith 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 $(\mathbb{x}, \mathbb{y}) \in L(\mathcal{R})$.

— Doomed transcript during 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 jth polynomial $\widehat{f}_{i,j} \in \mathbb{F}^{\leq d_{i,j}}[X]$ has degree at most $d_{i,j}$. The verifier is then given as a message $(\widehat{f}_{i,j}(\mathbb{F}))_{j\in m_i}$ where $\widehat{f}_{i,j}(\mathbb{F})$ is the evaluation of $\widehat{f}_{i,j}$ over the entire field $\mathbb{F}$.

Completeness and soundness for a poly-IOPP are similar to those 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 toward 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 $\widehat{f}_{i,j}$ as coefficients, and simply say that the prover outputs a polynomial. Similarly, since the verifier has oracle access to $\widehat{f}_{i,j}$ evaluated over the entire field, we simply denote that it has direct oracle access to $\widehat{f}_{i,j}$.

We also use 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 a field $\mathbb{F}$, an evaluation domain $\mathcal{L} \subseteq \mathbb{F}$, and a degree $d \in \mathbb{N}$ is the set of evaluations over $\mathcal{L}$ of univariate polynomials (over $\mathbb{F}$) of degree less than d:

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

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 $f : \mathcal{L} \to \mathbb{F}$, we sometimes use $\widehat{f} \in \mathbb{F}^{<d}[X]$ 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 $f : \mathcal{L} \to \mathbb{F}$, $\mathsf{List}(f, d, \delta)$ 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 article.

- In Section 4.1, we describe a theorem of [20] 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 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 ([20], Theorem 1.5). *Let* $\mathcal{C} := \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *be a Reed–Solomon code with rate* $\rho := d/|\mathcal{L}|$*, and let* $\mathsf{B}^*(\rho) := \sqrt{\rho}$*. For every* $\delta \in (0, 1-\mathsf{B}^*(\rho))$ *and functions* $f_1 \ldots, f_m : \mathcal{L} \to \mathbb{F}$*, if*

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

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

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

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

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

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

While in the above theorem $\mathsf{B}^*(\rho) = \sqrt{(}\rho)$, we use $\mathsf{B}^*(\rho)$ throughout this article so that our theorems remain generic. Thus, any new advancements improving the above theorem can be generically plugged into our theorems.

4.2 Univariate Function Quotienting

We define the *quotient* of a univariate function.

Definition 4.2. Let $f : \mathcal{L} \to \mathbb{F}$ be a function, $S \subseteq \mathbb{F}$ be a set, and $\mathsf{Ans}, \mathsf{Fill} : S \to \mathbb{F}$ be functions. Let $\widehat{\mathsf{Ans}} \in \mathbb{F}^{<|S|}[X]$ be the (unique) polynomial with $\widehat{\mathsf{Ans}}(x) = \mathsf{Ans}(x)$ for every $x \in S$, and let $\widehat{V}_S \in \mathbb{F}^{<|S|+1}[X]$ be the unique non-zero polynomial with $\widehat{V}_S(x) = 0$ for every $x \in S$.

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

$$\forall\, x \in \mathcal{L},\ \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(x) := \begin{cases} \mathsf{Fill}(x) & x \in S, \\ \frac{f(x) - \widehat{\mathsf{Ans}}(x)}{\widehat{V}_S(x)} & \text{otherwise.} \end{cases}$$

(Note that S may or may not be contained or, indeed, intersect with $\mathcal{L}$.)

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 $\widehat{f} \in \mathbb{F}^{<d}[X]$ be a polynomial and $S \subseteq \mathbb{F}$ be a set, and let $\widehat{V}_S \in \mathbb{F}^{<|S|+1}[X]$ be the unique non-zero polynomial with $\widehat{V}_S(x) = 0$ for every $x \in S$. The ***polynomial quotient*** $\mathsf{PolyQuotient}(\widehat{f}, S) \in \mathbb{F}^{<d-|S|}[X]$ is defined as follows:

$$\mathsf{PolyQuotient}(\widehat{f}, S)(X) := \frac{\widehat{f}(X) - \widehat{\mathsf{Ans}}(X)}{\widehat{V}_S(X)}\ ,$$

where $\widehat{\mathsf{Ans}} \in \mathbb{F}^{<|S|}[X]$ is the unique (nonzero) polynomial with $\widehat{\mathsf{Ans}}(x) = \widehat{f}(x)$ for every $x \in S$.

The following lemma, implicit in prior works (e.g., [14, 23]), shows that if a function is "quotiented by the wrong value", then its quotient is far from low-degree.

Lemma 4.4. *Let $f : \mathcal{L} \to \mathbb{F}$ 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 $\mathsf{Ans}, \mathsf{Fill} : S \to \mathbb{F}$ 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 $T := \{x \in \mathcal{L} \cap S : \widehat{\mathsf{Ans}}(x) \neq f(x)\}$.

Proof. Let $g := \mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})$ and suppose toward 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 $\widehat{w}(X) = \widehat{V}_S(X) \cdot \hat{g}(X) + \widehat{\mathsf{Ans}}(X)$ where $\widehat{\mathsf{Ans}}$ and $\widehat{V}_S$ are defined as in Definition 4.2. Observe that $\deg(\widehat{w}) < d$ and that for every $x \in \mathcal{L} \setminus T$ where $\hat{g}(x) = g(x)$, we have

$$\widehat{w}(x) = \widehat{V}_S(x) \cdot \hat{g}(x) + \widehat{\mathsf{Ans}}(x) = \widehat{V}_S(x) \cdot g(x) + \widehat{\mathsf{Ans}}(x) = f(x)\ .$$

The last equality follows by Definition 4.2 since:

- if $x \in S \setminus T$ then $\widehat{w}(x) = \widehat{\mathsf{Ans}}(x) = f(x)$, and
- if $x \notin S$ then $g(x) = \frac{f(x) - \widehat{\mathsf{Ans}}}{\widehat{V}_S(x)}$ so that $\widehat{V}_S(x) \cdot g(x) + \widehat{\mathsf{Ans}}(x) = f(x)$.

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 $\widehat{w}$ on $\mathcal{L}$ is δ-close to f. Moreover, for every $x \in S$ it holds that $\widehat{w}(x) = \widehat{\mathsf{Ans}}(x) = \mathsf{Ans}(x)$. 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 $f : \mathcal{L} \to \mathbb{F}$ 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*

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

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, it holds that $\Pr_{r \leftarrow \mathbb{F}\setminus\mathcal{L}}[\hat{u}(r) = \hat{u}'(r)] \le \frac{d-1}{|\mathbb{F}|-|\mathcal{L}|}$, so the probability that the polynomials agree on points $r_1, \ldots, r_s$ (which are independently sampled) 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 $r_1, \ldots, r_s \in \mathbb{F}$, there exist distinct codewords u, u' at distance at most δ from f such that $\hat{u}(r_i) = \hat{u}'(r_i)$ for all $i \in [s]$ 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., [14, 19, 23]).

The folding operator is based on the following fact, decomposing univariate polynomials into bivariate polynomials.

Fact 4.6 ([24]). *Given a polynomial $\hat{q} \in \mathbb{F}[X]$:*

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

We define the folding of a polynomial and then the folding of a function. When the function is the evaluation of a low-degree polynomial, these definitions coincide (on the evaluation domain).

Definition 4.7. Given a polynomial $\widehat{f} \in \mathbb{F}^{<d}[X]$, a folding parameter $k \in \mathbb{N}$, and $r \in \mathbb{F}$, we define a polynomial $\mathsf{PolyFold}(\widehat{f}, k, r) \in \mathbb{F}^{<d/k}[X]$ as follows: Let $\widehat{Q} \in \mathbb{F}[X,Y]$ be the bivariate polynomial derived from $\widehat{f}$ using Fact 4.6 with $\hat{q}(X) := X^k$. Then $\mathsf{PolyFold}(\widehat{f}, k, r)(X) := \widehat{Q}(X, r)$.

Definition 4.8. Let $f : \mathcal{L} \to \mathbb{F}$ 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 $\mathsf{Fold}(f, k, \alpha) : \mathcal{L}^k \to \mathbb{F}$ 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 $f : \mathcal{L} \to \mathbb{F}$, 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 $\rho := d/|\mathcal{L}|$,*

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

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

Proof. Suppose toward contradiction, that

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

Letting $\hat{p}_x$ be defined from f as in Definition 4.8, define $c_0,\ldots,c_{k-1}$ where $c_j : \mathcal{L}^k \to \mathbb{F}$ is the function where $c_j(x)$ is the jth coefficient of $\hat{p}_x$ (i.e., so that $\hat{p}_x(X) \equiv \sum_{j=0}^{k-1} c_j(x)\cdot X^j$ for every $x \in \mathcal{L}^k$). Observe that

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

Therefore, we get that

$$\begin{aligned}
&\Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta\left(\sum_{j=0}^{k-1} c_j\cdot(r^{\mathsf{fold}})^j, \mathsf{RS}[\mathbb{F},\mathcal{L}^k,d/k]\right) \le \delta\right] \\
&= \Pr_{r^{\mathsf{fold}} \leftarrow \mathbb{F}}\left[\Delta(\mathsf{Fold}(f,k,r^{\mathsf{fold}}), \mathsf{RS}[\mathbb{F},\mathcal{L}^k,d/k]) \le \delta\right] \\
&> \mathsf{err}^\star(d/k,\rho,\delta,k) \ .
\end{aligned}$$

By Theorem 4.1, there exists a set $S \subseteq \mathcal{L}^k$ with $|S| \ge (1-\delta)\cdot|\mathcal{L}^k|$ 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

$$\widehat{Q}(X,Y) := \sum_{x\in S'} I_{x,S'}(X)\cdot\hat{p}_x(Y) \ .$$

The partial degrees of $\widehat{Q}$ are $\deg_X(\widehat{Q}) < d/k$ and $\deg_Y(\widehat{Q}) < k$.

For every $\alpha \in \mathbb{F}$ and $x \in S'$, $\widehat{Q}(x,\alpha) = \hat{p}_x(\alpha) = \sum_{j=0}^{k-1} c_j(x)\cdot\alpha^j = \sum_{j=0}^{k-1}\hat{u}_j(x)\cdot\alpha^j$. If $|S'| = d/k$ then, since the degree of $\hat{u}_j$ is d/k, it holds that $\widehat{Q}(X,\alpha) \equiv \sum_{j=0}^{k-1}\hat{u}_j(X)\cdot\alpha^j$. 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 $\widehat{Q}(x,Y) \equiv \hat{p}_x(Y)$ for every $x \in S$. If $|S| < d/k$ then $S = S'$, and so this holds trivially.

Observe that the polynomial $\widehat{f}(X) := \widehat{Q}(X^k, X)$ has degree d. Moreover, by construction for every x with $x^k \in S$:

$$\widehat{f}(x) = \widehat{Q}(x^k,x) = \hat{p}_{x^k}(x) = f(x) \ .$$

Thus, there are at least $k \cdot |S| \geq k \cdot (1-\delta) \cdot |\mathcal{L}^k| = (1-\delta) \cdot |\mathcal{L}|$ points where $\widehat{f}$ 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 $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$, and degrees $0 \leq d_1, \ldots, d_m \leq d^*$, we define $\mathsf{Combine}(d^*, r, (f_1, d_1), \ldots, (f_m, d_m)) : \mathcal{L} \to \mathbb{F}$ 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 $f : \mathcal{L} \to \mathbb{F}$, and degree $0 \leq d \leq d^*$, we define $\mathsf{DegCor}(d^*, r, f, d) : \mathcal{L} \to \mathbb{F}$ 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 batch-testing of multiple polynomials of varying degrees can be done efficiently as long as the proximity error is bounded by $\min\{1 - \mathsf{B}^\star(\rho), 1 - \rho - 1/|\mathcal{L}|\}$ where $\rho = d^*/|\mathcal{L}|$.

Lemma 4.13. *Let $d^* \in \mathbb{N}$ be a target degree, $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$ 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*

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

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

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

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,

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

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

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

By Theorem 4.1, applied on the functions $f_{i,j} = x^\ell f_i$ and noting that the r values form the correct curve, there exists a set $S \subseteq \mathcal{L}$ with $|S| > (1-\delta)\cdot|\mathcal{L}|$ such that for every $i \in [m]$ and $\ell \in \{0, \ldots, d^*-d_i\}$ there exists a polynomial $\hat{p}_{i,\ell} \in \mathbb{F}^{<d^*}[X]$ such that $\hat{p}_{i,\ell}(x) = x^\ell \cdot f_i(x)$ for every $x \in S$.

Fix $i \in [m]$. We inductively show that $\deg(\hat{p}_{i,\ell}) < d_i + \ell$. 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 (so that the function $u_i \in \mathsf{RS}[\mathbb{F}, \mathcal{L}d_i]$ defined as $u_i(x) = \hat{p}_i(x)$ for all $x \in \mathcal{L}$ is the codeword promised in the lemma), and this was true for any $i \in [m]$.

As the base case, it is immediate that $\deg(\hat{p}_{i,d^*-d_i}) < d^* = d_i + d^* - d_i$. For $0 \le \ell < d^* - d_i$ suppose that $\deg(\hat{p}_{i,\ell+1}) < d_i + \ell + 1$. We show that $\deg(\hat{p}_{i,\ell}) < d_i + \ell$. Consider the polynomial $\hat{q}(X) := X \cdot \hat{p}_{i,\ell}(X)$. Since $\deg(\hat{p}_{i,\ell}) < d^*$, it follows that $\deg(\hat{q}) < d^* + 1$. Observe that for every $x \in S$,

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

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

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* $k \ge 4$.

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 $f_0 : \mathcal{L}_0 \to \mathbb{F}$ 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 $\widehat{f}_0 \in \mathbb{F}^{<d_0}[X]$ whose restriction to $\mathcal{L}_0$ is f_0.
— **Initial folding:** The verifier sends $r_0^{\text{fold}} \leftarrow \mathbb{F}$.
— **Interaction phase loop:** For $i = 1, \ldots, M$:
(1) **Send the folded function:** The prover sends a function $g_i : \mathcal{L}_i \to \mathbb{F}$. In the honest case, g_i is the evaluation of the polynomial $\hat{g}_i := \mathsf{PolyFold}(\widehat{f}_{i-1}, k_{i-1}, r_{i-1}^{\text{fold}})$ over $\mathcal{L}_i$ (see Definition 4.7 to recall the definition).
(2) **Out-of-domain samples:** The verifier sends $r_{i,1}^{\text{out}}, \ldots, r_{i,s}^{\text{out}} \leftarrow \mathbb{F} \setminus \mathcal{L}_i$.
(3) **Out-of-domain reply:** The prover sends field elements $\beta_{i,1}, \ldots, \beta_{i,s} \in \mathbb{F}$. In the honest case, $\beta_{i,j} := \hat{g}_i(r_{i,j}^{\text{out}})$.
(4) **STIR message:** The verifier sends $r_i^{\text{fold}}, r_i^{\text{comb}} \leftarrow \mathbb{F}$ and $r_{i,1}^{\text{shift}}, \ldots, r_{i,t_{i-1}}^{\text{shift}} \leftarrow \mathcal{L}_{i-1}^{k_{i-1}}$.
(5) **Define next polynomial and send hole fills:** The prover sends oracle message

$$\mathsf{Fill}_i : \{r_{i,j}^{\text{shift}} : j \in [t_{i-1}]\} \cap \mathcal{L}_i \to \mathbb{F}.$$

In the honest case, the prover defines

$$\begin{aligned} \mathcal{G}_i &:= \{r_{i,j}^{\text{out}} : j \in [s]\} \\ &\quad \cup \{r_{i,j}^{\text{shift}} : j \in [t_{i-1}]\}, \\ \hat{g}_i' &:= \mathsf{PolyQuotient}(\hat{g}_i, \mathcal{G}_i), \end{aligned}$$

and $\mathsf{Fill}_i(r_{i,j}^{\text{shift}}) := \hat{g}_i'(r_{i,j}^{\text{shift}})$ if $r_{i,j}^{\text{shift}} \in \mathcal{L}_i$.

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

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

The protocol proceeds to the next iteration with $\widehat{f}_i$.

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

— **Final round:** The prover sends d_M coefficients of a polynomial $\hat{p} \in \mathbb{F}^{<d_M}[X]$. In the honest case, $\hat{p} := \mathsf{Fold}(\widehat{f}_M, k_M, r_M^{\mathsf{fold}})$.
— **Verifier decision phase**:
(1) **Main loop:** For $i = 1, \ldots, M$:
(a) For every $j \in [t_{i-1}]$, query $\mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}})$ at $r_{i,j}^{\mathsf{shift}}$. This involves querying f_{i-1} at all k_{i-1} points $x \in \mathcal{L}_{i-1}$ with $x^{k_{i-1}} = r_{i,j}^{\mathsf{shift}}$.
(b) Define $\mathcal{G}_i := \{r_{i,1}^{\mathsf{out}}, \ldots, r_{i,s}^{\mathsf{out}}, r_{i,1}^{\mathsf{shift}}, \ldots, r_{i,t_{i-1}}^{\mathsf{shift}}\}$, and let $\mathsf{Ans}_i : \mathcal{G}_i \to \mathbb{F}$ be the function where $\mathsf{Ans}_i(r_{i,j}^{\mathsf{out}}) = \beta_{i,j}$ and $\mathsf{Ans}_i(r_{i,j}^{\mathsf{shift}}) = \mathsf{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\mathsf{fold}})(r_{i,j}^{\mathsf{shift}})$. Finally, (virtually) set $g_i' := \mathsf{Quotient}(g_i, \mathcal{G}_i, \mathsf{Ans}_i, \mathsf{Fill}_i)$.
(c) Define the virtual oracle $f_i : \mathcal{L}_i \to \mathbb{F}$ as follows:

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

Observe that a query x to f_i translates to a single query either to g_i (if $x \notin \mathcal{G}_i$) or to Fill_i (if $x \in \mathcal{G}_i$).
(2) **Consistency with final polynomial:**
(a) Sample random points $r_1^{\mathsf{fin}}, \ldots, r_{t_M}^{\mathsf{fin}} \leftarrow \mathcal{L}_M^{k_M}$.
(b) Check that $\hat{p}(r_j^{\mathsf{fin}}) = \mathsf{Fold}(f_M, k_M, r_M^{\mathsf{fold}})(r_j^{\mathsf{fin}})$ for every $j \in [t_M]$.
(3) **Consistency with** Ans**:** For every $i \in \{1, \ldots, M\}$ and every $x \in \mathcal{G}_i \cap \mathcal{L}_i$ query $g_i(x)$ and check that $g_i(x) = \mathsf{Ans}_i(x)$.

Remark 5.3. If $\mathcal{L}_{i-1}^{k_{i-1}} \cap \mathcal{L}_i = \emptyset$ for every $i \in [M]$, then the oracles Fill_i and the verifier's check in Item 5.2 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}. 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 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 the parameters* $\mathbb{F}$, M, d, s, $(k_i)_{i=0}^M$, $(\mathcal{L}_i)_{i=0}^M$, $(t_i)_{i=0}^M$, *and* $(d_i)_{i=0}^M$ *as in Construction 5.2, and for every* $0 \le i \le M$ *let* $\rho_i := d_i/|\mathcal{L}_i|$. *For every* $f \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}_0, d_0]$ *and every* $\delta_0, \ldots, \delta_M$ *where*

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

STIR (Construction 5.2) has round-by-round soundness error $(\varepsilon^{\mathsf{fold}}, (\varepsilon_i^{\mathsf{out}}, \varepsilon_i^{\mathsf{shift}})_{i \in [M]}, \varepsilon^{\mathsf{fin}})$ *where:*

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

Above, B^* *and* err^* *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 $\mathsf{tr}_j := ((r_{j,1}^{\mathsf{out}} \ldots, r_{j,s}^{\mathsf{out}}), (\beta_{j,1}, \ldots, \beta_{j,s_i}), (r_j^{\mathsf{fold}}, r_j^{\mathsf{comb}}, r_{j,1}^{\mathsf{shift}}, \ldots, r_{j,t_j}^{\mathsf{shift}}))$ 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 5.2) for $i-1$ times.

We now describe the state function State and analyze its error.

(0) **State function for empty transcript.** Given a function $f : \mathcal{L} \to \mathbb{F}$ 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

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

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

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

(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

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

Suppose that $\mathsf{State}(f, \mathsf{tr}) = 0$. If there exists $j \in \{1, \ldots, i-1\}$ and $x \in \mathcal{L}_j \cap \mathcal{G}_j$ such that $g_j(x) \neq \mathsf{Ans}_j(x)$ then Item 5.2 cannot hold. Thus, in order for the state to switch to 1, we assume Item 5.2 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 5.2. Hence, we only need to bound the probability that Item 5.2 holds; by Lemma 4.5 this probability is at most $\frac{\ell_i^2}{2} \cdot \left(\frac{d_i}{|\mathbb{F}| - |\mathcal{L}_i|} \right)^s$.

(3) **Bounding $\varepsilon_i^{\mathsf{shift}}$.** The transcript so far has the form $\mathsf{tr} := (r_0^{\mathsf{fold}}, \mathsf{tr}_1, \ldots, \mathsf{tr}_{i-1}, g_i, r_i^{\mathsf{out}}, \beta_i)$ 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 5.2 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 (Definition 5.2).

— *Bounding the error.* We show that

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

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 5.2 does not hold), then Item 5.2 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 5.2 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 $\hat{u}(r_{i,j}^{\text{shift}}) \neq \text{Fold}(f_{i-1}, k_{i-1}, r_{i-1}^{\text{fold}})(r_{i,j}^{\text{shift}}) = \text{Ans}_i(r_{i,j}^{\text{shift}})$ 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 the event described in Claim 5.5 occurs. Then by Lemma 4.4:

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

Due to Item 5.2, in order for the state to become 1, it must be that $g_i(x) = \text{Ans}_i(x)$ for every $x \in \mathcal{L}_i \cap \mathcal{G}_i$, and so we conclude that $\Delta\left(g_i', \text{RS}[\mathbb{F}, \mathcal{L}_i, d_i - |\mathcal{G}_i|]\right) > \delta_i$. Recalling that $\delta_i < \min\{1 - \mathsf{B}^*(\rho_i), 1 - \rho_i - 1/|\mathcal{L}_i|\}$ and using Lemma 4.13 we deduce that

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

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

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

Putting all of the above probabilities together, we conclude that

$$\varepsilon_i^{\text{shift}} \leq (1-\delta_{i-1})^{t_{i-1}} + \text{err}^*(d_i, \rho_i, \delta_i, t_{i-1} + s) + \text{err}^*(d_i/k_i, \rho_i, \delta_i, k_i) \ .$$

(1) **Bounding ε^{fin}:** At this stage, the transcript holds the form $\text{tr} := (r_0^{\text{fold}}, \text{tr}_1, \ldots, \text{tr}_M, \hat{p})$. The verifier chooses points $(r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})$. Deriving f_M from the transcript, we define the state function.
— *State function.* We set $\text{State}(f, \text{tr} \| (r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})) = 1$ if and only if both of the following hold:
  (a) $\hat{p}(r_j^{\text{fin}}) = \text{Fold}(f_M, k, r_M^{\text{fold}})(r_j^{\text{fin}})$ for every $j \in [t_i]$.
  (b) For every $0 < j \leq M$: $g_j(x) = \text{Ans}_j(x)$ for every $x \in \mathcal{L}_j \cap \mathcal{G}_j$ where Ans_j is defined as in the verifier decision algorithm.
— *Bounding the error.* We show that

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

Suppose that $\text{State}(f, \text{tr}) = 0$. If Item 5.2 does not hold, then Item 1b also does not hold, and so the state cannot change to 1. Assuming this is not the case, it follows from Item 5.2 that:

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

Thus, for each j the probability that $\hat{p}(r_j^{\text{fin}}) = \text{Fold}(f_M, k_M, r_M^{\text{fold}})(r_j^{\text{fin}})$ is at most $1 - \delta_M$. It follows that the probability that this occurs for every $j \in [t_M]$ simultaneously is at most $(1-\delta_M)^{t_M}$.

(2) **Verifier decision.** If $\text{State}(f, \text{tr}) = 0$ for a full transcript tr, then the verifier rejects. This is due to the fact that the verifier checks in Item 5.2 that Item 1a holds, and in Item 5.2 that Item 1b holds. If $\text{State}(f, \text{tr}) = 0$ 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 Section A.

Ingredients. In both settings, the following ingredients are fixed: (a) a Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ with rate $\rho := d/|\mathcal{L}|$ where $d \geq 4$ is a power of two, and $\mathcal{L}$ is a smooth domain, (b) a security parameter $\lambda \in \mathbb{N}$, (c) a stopping degree $d_{\text{stop}} \in \mathbb{N}$ where d_{stop} is a power of two such that $d \geq d_{\text{stop}} \geq 4$, (d) a proximity parameter $\delta \in (0, 1)$, and (e) a folding parameter $k \in \mathbb{N}$ that is a power of two where $k \geq 4$.

5.3.1 Provable Soundness. In the provable soundness regime, we additionally set the following quantities:

- $M := \left\lfloor \log_k(d/d_{\text{stop}}) \right\rfloor$, $s := 1$;
- $\mathcal{L}_i = \omega \cdot \langle \omega^2 \rangle$ where ω is a generator of $\mathcal{L}_{i-1}$; [8]
- $\eta_0 = \left(\frac{2^\lambda \cdot (k-1) \cdot (d/k)^2}{2^7 \cdot |\mathbb{F}|} \right)^{1/7}$, $t_0 := \left\lceil \frac{\lambda+1}{-\log(1-\delta')} \right\rceil$ where $\delta' := \min\{\delta, 1 - \sqrt{\rho} - \eta_0\}$;
- for $0 < i < M$ set $k_i := k$, $d_i := \frac{d}{\prod_{j<i} k_j}$, $\rho_i := (2/k)^i \cdot \rho$,

$$\eta_i := \max\left\{ \left(\frac{2^\lambda \cdot d_i}{8 \cdot \rho_i \cdot (|\mathbb{F}| - |\mathcal{L}_i|)} \right)^{1/2}, \left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2 \right)}{2^7 \cdot |\mathbb{F}|} \right)^{1/7} \right\},$$

  and $t_i := \left\lceil \frac{\lambda+1}{-\log(\sqrt{\rho_i} + \eta_i)} \right\rceil$; and
- $t_M := \left\lceil \frac{\lambda}{-\log(\sqrt{\rho_M} + \eta_M)} \right\rceil$.

Note that the η_i values defined above are the slack parameters that inform the distance from the Johnson bound (i.e., $\delta_i = 1 - \sqrt{\rho_i} - \eta_i$).

Resulting IOPP. We instantiate Construction 5.2 with:

- field $\mathbb{F}$;
- iteration count $M \in \mathbb{N}$;
- initial degree parameter $d \in \mathbb{N}$;
- folding parameters $k_0, \ldots, k_M \in \mathbb{N}$;
- evaluation domains $\mathcal{L}_0, \ldots, \mathcal{L}_M \subseteq \mathbb{F}$;
- repetition parameters $t_0, \ldots, t_M \in \mathbb{N}$; and
- out-of-domain repetition parameter $s \in \mathbb{N}$.

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:

[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 Lemma 5.3.
[9] Note that this bound is not tight.

— *Rounds:* $2 \cdot M + 1$.
— *Length:* $|\mathcal{L}| + O_k(d_{\text{stop}} + \log(d/d_{\text{stop}}))$.
— *Input queries:* $\left\lceil \frac{\lambda}{-\log(1-\delta')} \right\rceil$ where $\delta' := \min\{\delta, 1 - 1.05 \cdot \sqrt{\rho}\}$.
— *Proof queries:* $O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{-\log \rho}\right)\right)$.
— *Round-by-round soundness error:* $2^{-\lambda}$.

See Section A.1 for a detailed derivation of these values.

5.3.2 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 ([20, 23]). *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 $f_1 \ldots, f_m : \mathcal{L} \to \mathbb{F}$, if*

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

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

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

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

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 Section A.2 for full derivations given this parameter setting.

6 Implementation and Experimental Results

We evaluate the performance of STIR in comparison to FRI [19], 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` [11] 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` [11] 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.

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.

Repetitions. The number of repetitions in FRI and in STIR to achieve round-by-round soundness error $2^{-\lambda_{\text{prot}}}$ was selected by assuming 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 [22], 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 [31] for the Merkle trees.

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

When used to instantiate a SNARK, 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. The full experimental results can be found in the full version [12]. We discuss each metric individually, arbitrarily focusing on the case of degree $d = 2^{24}$ and rate $\rho = 1/2$.

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 $2,645$ hashes and runs in 3.8ms whereas the FRI verifier performs $5,647$ 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. Note that, as the prover is implemented with a proof of work to enhance security, there is

[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.

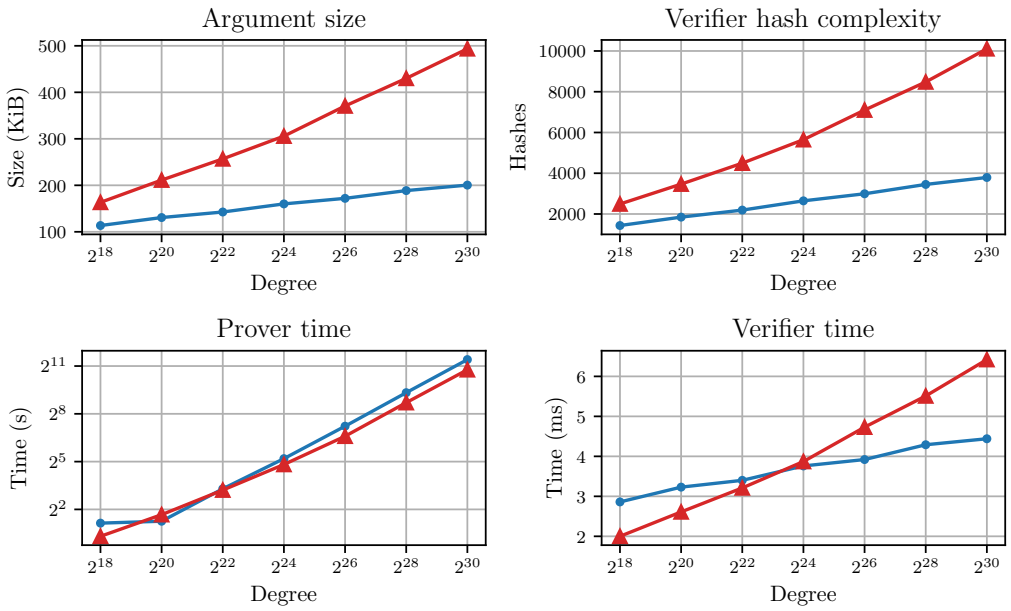


Fig. 2. Comparison of FRI and STIR for $\rho = 1/2$. FRI: s, STIR: $\bullet$. Lower is better. Note that prover time is displayed with logarithmic scaling.

high variance in prover times for small values of d (this is true for both FRI and STIR). Therefore, for small degrees, these results should be taken with a grain of salt.

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. In our experiments, since the relative cost of performing hashes is higher in this configuration, we observe that STIR's verifier now outperforms FRI's across all settings tested. We measure this speedup to be between 1.48× to 2.34×.

SNARK for R1CS. We computed the argument size for both STIR and FRI when used to instantiate a SNARK for R1CS using an idealized Python script. We used the *native* configuration for both. Details on the SNARK constructing are in the full version of this work [12]. 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.

Table 3. Comparison of STIR and FRI on Different Rates

$d = 2^{24}$			
	STIR $\rho = 1/4$	FRI $\rho = 1/8$	Ratio
Argument size	107 KiB	134 KiB	1.25×
Verifier time	2.4 ms	1.8 ms	0.75×
Verifier hashes	1,849	2,720	1.47×
Prover time	58 s	93 s	1.60×
$d = 2^{28}$			
	STIR $\rho = 1/4$	FRI $\rho = 1/8$	Ratio
Argument size	128 KiB	184 KiB	1.44×
Verifier time	2.8 ms	2.4 ms	0.85×
Verifier hashes	2401	3879	1.61×
Prover time	1100 s	1700 s	1.54×

— 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 tradeoff 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.

Acknowledgements

We thank Ariel Gabizon, Ron Rothblum, and Justin Thaler for helpful comments on presentation, and Marcin Górny, Francesco Intoci, Pratyush Mishra, and Andrew Zitek-Estrada for assisting with the Merkle tree and Fast Fourier Transform implementations in `arkworks`.

References

[1] 2024. Hermez. Retrieved October 24, 2026 from https://hermez.io
[2] 2024. Miden. Retrieved October 24, 2026 from https://github.com/0xPolygonMiden
[3] 2024. Neptune. Retrieved October 24, 2026 from https://neptune.cash/
[4] 2024. Ola. Retrieved October 24, 2026 from https://ola.finance
[5] 2024. Polygon. Retrieved October 24, 2026 from https://polygon.technology
[6] 2024. Risc0. Retrieved October 24, 2026 from https://risc0.com
[7] 2024. Sandstorm. Retrieved October 24, 2026 from https://github.com/andrewmilson/sandstorm
[8] 2024. StarkEx. Retrieved October 24, 2026 from https://starkware.co/starkex/
[9] 2024. StarkNet. Retrieved October 24, 2026 from https://www.starknet.io/
[10] 2024. zkSync. Retrieved October 24, 2026 from https://zksync.io
[11] arkworks contributors. 2024. arkworks: An Ecosystem for Developing and Programming with zkSNARKs. Rust libraries. Retrieved October 24, 2026 from https://arkworks.rs
[12] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. 2024. STIR: Reed–solomon proximity testing with fewer queries. In *Proceedings of the 44th Annual International Cryptology Conference*. 380–413. Retrieved from https://eprint.iacr.org/2024/390

[13] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. 2025. WHIR: Reed-solomon proximity testing with super-fast verification. In *Proceedings of the 44th Annual International Conference on Theory and Application of Cryptographic Techniques*. 214–243.
[14] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. 2023. IOPs with inverse polynomial soundness error. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science*. IEEE, 752–761.
[15] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof verification and the hardness of approximation problems. *Journal of the ACM* 45, 3 (1998), 501–555.
[16] Sanjeev Arora and Shmuel Safra. 1998. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM* 45, 1 (1998), 70–122.
[17] László Babai. 1985. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. 421–429.
[18] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. 1991. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. 21–32.
[19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Fast Reed–Solomon interactive oracle proofs of proximity. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. 14:1–14:17.
[20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. 2020. Proximity gaps for Reed–Solomon codes. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. 900–909.
[21] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. 2019. Aurora: Transparent succinct arguments for R1CS. In *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 103–128.
[22] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. 2016. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*. 31–60.
[23] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. 2020. DEEP-FRI: Sampling outside the box improves soundness. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. 5:1–5:32.
[24] Eli Ben-Sasson and Madhu Sudan. 2008. Short PCPs with polylog query complexity. *SIAM Journal on Computing* 38, 2 (2008), 551–607.
[25] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. 2023. Fiat-Shamir security of FRI and related SNARKs. In *Proceedings of the 29th International Conference on the Theory and Application of Cryptology and Information Security*. Jian Guo and Ron Steinfeld (Eds.), Lecture Notes in Computer Science, Vol. 14439, Springer, 3–40.
[26] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. 2025. Blaze: Fast SNARKs from interleaved RAA codes. In *Proceedings of the 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Serge Fehr and Pierre-Alain Fouque (Eds.), Lecture Notes in Computer Science, Vol. 15604, Springer, 123–152.
[27] Irit Dinur. 2007. The PCP theorem by gap amplification. *Journal of the ACM* 54, 3 (2007), 12.
[28] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. 1996. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM* 43, 2 (1996), 268–292.
[29] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1989. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* 18, 1 (1989), 186–208.
[30] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. 2023. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *Proceedings of the 43rd Annual International Cryptology Conference*.
[31] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. 2019. Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems. IACR Cryptology ePrint Archive, Report 2019/458.
[32] Yael Kalai and Ran Raz. 2008. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*. 536–547.
[33] Thilo Mie. 2009. Short PCPPs verifiable in polylogarithmic time with O(1) queries. *Annals of Mathematics and Artificial Intelligence* 56, 3 (2009), 313–338.
[34] I. S. Reed and G. Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics* 8, 2 (1960), 300–304.
[35] Omer Reingold, Ron Rothblum, and Guy Rothblum. 2016. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*. 49–62.
[36] Noga Ron-Zewi and Ron Rothblum. 2020. Local proofs approaching the witness length. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. 846–857.
[37] Noga Ron-Zewi and Ron D. Rothblum. 2022. Proving as fast as computing: succinct arguments with constant prover overhead. In *Proceedings of the 54th ACM Symposium on the Theory of Computing*. 1353–1363.

[38] StarkWare. 2021. ethSTARK Documentation. Cryptology ePrint Archive, Paper 2021/582. Retrieved October 24, 2026 from https://eprint.iacr.org/2021/582

[39] Hadas Zeilberger, Binyi Chen, and Ben Fisch. 2024. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *Proceedings of the 44th Annual International Cryptology Conference*. 138–169.

Appendix

A Derivations for Section 5.3

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

— In Section A.1, we give derivations for computing provable security bounds.
— In Section A.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 A.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 A.2. $\rho_i \geq \rho/d$.

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

A.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:

$$\begin{aligned}\left(\frac{2^{\lambda+1} \cdot \left(t_{i-1} \cdot d_i^2 + (k-1) \cdot (d_i/k)^2\right)}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} &< \left(\frac{2^{\lambda+1} \cdot (t_{i-1}+1) \cdot d^2}{2^7 \cdot |\mathbb{F}|}\right)^{1/7} \\ &< \left(\frac{2^{\lambda+1} \cdot (t_{i-1}+1) \cdot d^2}{2^7 \cdot 10^7 \cdot 2^{\lambda+1} \cdot d^2 \cdot |\mathcal{L}|^{3.5} \cdot (t_{i-1}+1)}\right)^{1/7} \\ &< \left(\frac{1}{20^7 \cdot |\mathcal{L}|^{3.5}}\right)^{1/7} \\ &= \frac{\sqrt{\rho/d}}{20} \ .\end{aligned}$$

Complexity parameters. The complexity parameters of the protocol are as follows:

— *Rounds.* $2M + 1 = 2 \cdot \left\lfloor \log_k(d/d_{\text{stop}}) \right\rfloor + 1$.

— *Proof length.* $M \cdot s + \frac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} |\mathcal{L}_i|$

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

— *Input query complexity.* $t_0 \leq \frac{\lambda}{-\log(1-\delta')}$.

— *Proof query complexity.* Observe that $\frac{\log(1.05\cdot\sqrt{\rho})}{\log(\sqrt{k/2})} < 0$ 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 A.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:

– $\varepsilon^{\text{fold}}$:

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

where the final inequality holds since $\eta_0 = \left(\frac{2^\lambda\cdot(k-1)\cdot(d/k)^2}{2^7\cdot|\mathbb{F}|}\right)^{1/7}$.

– $\varepsilon_i^{\text{out}}$:

$$\varepsilon_i^{\text{out}} \leq \frac{d_i^s\cdot t_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s} \leq \frac{1}{4\cdot\eta_i^2\cdot\rho_i}\cdot\frac{d_i}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)} = \frac{1}{\eta_i^2}\cdot\frac{d_i}{8\cdot\rho_i\cdot(|\mathbb{F}|-|\mathcal{L}_i|)} \leq 2^{-\lambda} \ ,$$

where the final inequality holds since $\eta_i \geq \left(\frac{2^\lambda\cdot d_i}{8\cdot\rho_i\cdot(|\mathbb{F}|-|\mathcal{L}_i|)}\right)^{1/2}$.

– $\varepsilon_i^{\text{shift}}$: 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:

$$\begin{aligned}
&\text{err}^\star(d_i, \rho_i, \delta_i, t_{i-1}+s) + \text{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \\
&\qquad = \frac{t_{i-1}\cdot d_i^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{\eta_i, \frac{\sqrt{\rho_i}}{20}\right\}\right)^7} + \frac{(k-1)\cdot(d_i/k)^2}{|\mathbb{F}|\cdot\left(2\cdot\min\left\{\eta_i, \frac{\sqrt{\rho_i}}{20}\right\}\right)^7} \\
&\qquad = \frac{t_{i-1}\cdot d_i^2 + (k-1)\cdot(d_i/k)^2}{|\mathbb{F}|\cdot(2\cdot\eta_i)^7} \\
&\qquad = 2^{-\lambda-1} \ .
\end{aligned}$$

The final inequality holds since $\eta_i \geq \left(\frac{2^{\lambda+1}\cdot(t_{i-1}\cdot d_i^2+(k-1)\cdot(d_i/k)^2)}{2^7\cdot|\mathbb{F}|}\right)^{1/7}$. Finally,

$$\begin{aligned}
\varepsilon_i^{\text{shift}} &\leq (1-\delta_{i-1})^{t_{i-1}} + \text{err}^\star(d_i, \rho_i, \delta_i, t_{i-1}+s) + \text{err}^\star(d_i/k_i, \rho_i, \delta_i, k_i) \\
&\leq 2^{-\lambda-1} + 2^{-\lambda-1} \\
&= 2^{-\lambda} \ .
\end{aligned}$$

– ε^{fin}: it holds that $\varepsilon^{\text{fin}} \leq (1-\delta_M)^{t_M} = (\sqrt{\rho_M}+\eta_M)^{\left\lceil\frac{\lambda}{-\log(\sqrt{\rho_M}+\eta_M)}\right\rceil} \leq 2^{-\lambda}$.

A.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

$$|\mathbb{F}| > (\lambda+1)\cdot 2^{\lambda+2}\cdot d\cdot|\mathcal{L}|^3\cdot\left(s+\max\left\{\left\lceil\frac{1}{-\log(1-\delta')}\right\rceil,\left\lceil\frac{1}{-\log(1.5\cdot\rho)}\right\rceil\right\}\right) \quad ,$$

it holds that $\eta_i \leq \rho_i/2$ for every i. First observe that

$$|\mathbb{F}| > 2^{\lambda+2}\cdot d\cdot|\mathcal{L}|^3\cdot(s+\max\{t_0,t_i\}) \quad ,$$

We now bound the η parameters.

— $i=0$:

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

— $i>0$:

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

$$= \max\left\{\frac{2\cdot\rho_i}{d_i},\frac{d_i}{\rho_i}\cdot\left(\frac{2^\lambda\cdot d_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^2}\right)^{\frac{1}{2}},\frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+1)+\left(\frac{k-1}{k}\right)\right)\right\} \quad .$$

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 $4\leq d_{\text{stop}}\leq d_i$.

– Second option:

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

– Third option:

$$\begin{aligned}\frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+1)+\left(\frac{k-1}{k}\right)\right) &< \frac{2^{\lambda+1}\cdot d_i}{\rho_i^2\cdot|\mathbb{F}|}\cdot(t_{i-1}+2)\\ &< \frac{2^{\lambda+1}\cdot d\cdot|\mathcal{L}|^2\cdot(t_{i-1}+2)}{2^{\lambda+2}\cdot d\cdot|\mathcal{L}|^3\cdot(t_{i-1}+2)}\\ &= \frac{\rho}{2\cdot d}\\ &< \frac{\rho_i}{2} \quad .\end{aligned}$$

Complexity parameters. The complexity parameters of the protocol are as follows:

— *Rounds.* $2M+1 = 2\cdot\left\lfloor\log_k(d/d_{\text{stop}})\right\rfloor+1$.

— *Proof length.* $M \cdot s + \dfrac{d}{\prod_{i=0}^{M} k_i} + \sum_{i=1}^{M} |\mathcal{L}_i|$

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

— *Input query complexity.* $t_0 \leq \frac{\lambda}{-\log(1-\delta')}$.

— *Proof query complexity.* Observe that $\frac{\log(1.5\cdot\rho)}{\log(k/2)} < 0$ since $\rho \leq 1/2$. Therefore the proof query complexity is:

$$\begin{aligned} \sum_{i=1}^{M} t_i &= \sum_{i=1}^{M} \left\lceil \frac{\lambda + 1}{-\log(\rho_i + \eta_i)} \right\rceil \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{-\log(\rho_i + \rho_i/2)} \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{-\log((2/k)^i \cdot 1.5 \cdot \rho)} \\ &\leq M + (\lambda+1) \cdot \sum_{i=1}^{M} \frac{1}{i \cdot \log(k/2) - \log(1.5 \cdot \rho)} \\ &\leq M + \frac{\lambda+1}{\log(k/2)} \cdot \sum_{i=1}^{M} \frac{1}{i - \frac{\log(1.5\cdot\rho)}{\log(k/2)}} \\ &< M + \frac{\lambda+1}{\log(k/2)} \cdot \left(\log\left(\frac{M}{-\frac{\log(1.5\cdot\rho)}{\log(k/2)}} + 1 \right) + 1 \right) \\ &= O_k\left(\log d + \lambda \cdot \log\left(\frac{\log d}{-\log \rho} \right) \right) \ , \end{aligned}$$

where the final inequality follows by applying Fact A.1.

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

— $\delta_0 \in (0, \Delta(f, \mathrm{RS}[\mathbb{F}, \mathcal{L}_0, d_0])] \cap (0, 1-\rho_0)$: this holds by the definition of $\delta_0 := \min\{\delta, 1 - \rho_0 - \eta_0\}$, since $\delta := \Delta(f, \mathrm{RS}[\mathbb{F}, \mathcal{L}_0, d_0])$ 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 $\eta_i \geq 2/|\mathcal{L}_i|$.
— $\mathrm{RS}[\mathbb{F}, \mathcal{L}_i, d_i]$ 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

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

as in Conjecture 5.6:

— $\varepsilon^{\text{fold}}$:

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

where the final inequality holds since $\eta_0 = \left(\frac{2^{\lambda}\cdot(k-1)^{c_2}\cdot(d/k)^{c_2}}{\rho^{c_1+c_2}\cdot|\mathbb{F}|}\right)^{1/c_1}$.

— $\varepsilon_i^{\text{out}}$:

$$\varepsilon_i^{\text{out}} \leq \frac{d_i^s\cdot\ell_i^2}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s} \leq \left(\frac{d_i}{\rho_i\cdot\eta_i}\right)^{2\cdot c_3}\cdot\frac{d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s} \leq 2^{-\lambda}\ ,$$

where the final inequality holds since $\eta_i \geq \frac{d_i}{\rho_i}\cdot\left(\frac{2^{\lambda}\cdot d_i^s}{2\cdot(|\mathbb{F}|-|\mathcal{L}_i|)^s}\right)^{\frac{1}{2\cdot c_3}}$.

— $\varepsilon_i^{\text{shift}}$: 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:

$$\begin{aligned}&\mathsf{err}^{\star}(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^{\star}(d_i/k_i, \rho_i, \delta_i, k_i)\\ &\quad= \frac{(t_{i-1}+s-1)^{c_2}\cdot d_i^{c_2}}{\eta_i^{c_1}\cdot\rho_i^{c_1+c_2}\cdot|\mathbb{F}|} + \frac{(k-1)^{c_2}\cdot(d_i/k)^{c_2}}{\eta_i^{c_1}\cdot\rho_i^{c_1+c_2}\cdot|\mathbb{F}|}\\ &\quad= \frac{d_i^{c_2}}{\eta_i^{c_1}\cdot\rho_i^{c_1+c_2}\cdot|\mathbb{F}|}\cdot\left((t_{i-1}+s-1)^{c_2} + \left(\frac{k-1}{k}\right)^{c_2}\right)\\ &\quad= 2^{-\lambda-1}\ .\end{aligned}$$

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,

$$\begin{aligned}\varepsilon_i^{\text{shift}} &\leq (1-\delta_{i-1})^{t_{i-1}} + \mathsf{err}^{\star}(d_i, \rho_i, \delta_i, t_{i-1}+s) + \mathsf{err}^{\star}(d_i/k_i, \rho_i, \delta_i, k_i)\\ &\leq 2^{-\lambda-1} + 2^{-\lambda-1}\\ &= 2^{-\lambda}\ .\end{aligned}$$

— ε^{fin}: it holds that $\varepsilon^{\text{fin}} \leq (1-\delta_M)^{t_M} = (\rho_M+\eta_M)^{\left\lceil\frac{\lambda}{-\log(\rho_M+\eta_M)}\right\rceil} \leq 2^{-\lambda}$.

Received 26 March 2025; revised 28 February 2026; accepted 2 October 2026