University of Tartu
Faculty of Science and Technology
Institute of Mathematics and Statistics

Martin Puškin

An Analysis of the WHIR Interactive Oracle Proof of Proximity

Mathematics

Master Thesis (30 ECTS)

Supervisor:
prof. Helger Lipmaa

Tartu 2026

AN ANALYSIS OF THE WHIR INTERACTIVE ORACLE PROOF OF PROXIMITY

Master thesis

Martin Puškin

Abstract

Interactive oracle proofs of proximity (IOPPs) play a central role in the construction of modern hash-based zk-SNARKs. Among these, the WHIR protocol introduced in 2024 is one of the most verifier-efficient known IOPPs for testing proximity to multilinear Reed–Solomon codes. Due to its recent introduction and technical complexity, however, the protocol has received little detailed exposition beyond the original paper.

In this thesis, we present a comprehensive study of the WHIR protocol. Following the original work, we develop the necessary background on constrained Reed–Solomon codes and list decoding, and analyze key concepts such as folding, mutual correlated agreement, and block relative distance. We then introduce the WHIR protocol itself, analyze its asymptotic parameters, and prove its completeness and round-by-round soundness. Throughout, we clarify the design choices underlying WHIR and correct technical inaccuracies present in the original presentation.

Finally, we apply WHIR to transform a concrete polynomial interactive oracle proof into an interactive oracle proof, providing a fully worked example. Our aim is to make WHIR more accessible and to provide a clear and reliable reference for its use in hash-based zk-SNARK constructions.

CERCS research specialisation: P170 Computer science, numerical analysis, systems, control.

Key Words: zk-SNARKs, Reed–Solomon codes, WHIR, proofs of proximity.

ORAAKLIGA INTERAKTIIVSE LÄHEDUSTÕESTUSPROTOKOLLI WHIR ANALÜÜS

Magistritöö

Martin Puškin

Lühikokkuvõte

Oraakliga interaktiivsed lähedustõestused (interactive oracle proofs of proximity, IOPP-d) omavad keskset rolli kaasaegsete räsifunktsioonidel põhinevate zk-SNARKide konstrueerimisel. Nende seas on 2024. aastal tutvustatud WHIR üks verifitseerija seisukohast efektiivsemaid teadaolevaid IOPP-sid multilineaarsetele Reedi–Solomoni koodidele läheduse testimiseks. Oma hiljutise ilmumise ja tehnilise keerukuse tõttu on protokoll väljaspool algset artiklit seni pälvinud vähe detailset käsitlust.

Selles töös esitame põhjaliku WHIR-protokolli analüüsi. Järgime originaalartiklit, arendades välja vajaliku tagapõhja piirangutega Reedi–Solomoni koodide ja list-dekodeerimise kohta, ning analüüsime võtmemõisteid nagu voltimine, ühine korrelleeritud vastavus ja suhteline blokk-kaugus. Seejärel tutvustame WHIR-protokolli ennast, analüüsime selle asümptootilisi parameetreid ning tõestame selle täielikkuse ja voorupõhise veatuse. Läbivalt selgitame WHIR-protokolli konstruktsiooni aluseks olevaid disainivalikuid ning parandame algses esituses esinevaid tehnilisi ebatäpsusi.

Lõpetuseks teisendame konkreetse polünoomse interaktiivse oraakeltõestuse WHIR-protokolli abil interaktiivseks oraakeltõestuseks, esitades täielikult läbi töötatud näite. Töö eesmärk on muuta WHIR-protokoll paremini mõistetavaks ning pakkuda selget ja usaldusväärset viidet selle kasutamiseks räsifunktsioonidel põhinevate zk-SNARKide konstruktsioonides.

CERCS teaduseriala: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria).

Märksõnad: Nullteadmustõestused, Reedi–Solomoni koodid, WHIR, lähedustõestused.

Contents

1 Introduction **5**

2 Notation and preliminaries **9**
2.1 Multilinear polynomials 10
2.2 Schwartz–Zippel 12

3 Reed–Solomon Codes **13**
3.1 Error-correcting codes 13
3.2 Reed–Solomon Codes 21
3.3 Folding 24
3.4 Mutual correlated agreement 28
3.5 Block relative distance 36
3.6 Folding preserves list decoding 39
3.7 Out-of-domain sampling 44

4 IOPs and PIOPs **46**
4.1 IOPs and IOPPs 46
4.2 PIOPs 50
4.3 Getting a zk-SNARK from an IOP 52
4.3.1 Merkle trees 52
4.3.2 Fiat–Shamir 53
4.4 Compiling a PIOP into an IOP 53
4.5 Sumcheck 56

5 WHIR **58**
5.1 High level overview of WHIR 58
5.2 WHIR construction 61
5.2.1 WHIR parameters 65
5.3 WHIR is complete 67
5.4 WHIR is round-by-round sound 70
5.4.1 Notation 71
5.4.2 The state function 72
5.4.3 Bounding the errors 75
5.4.4 Choosing parameters 81
5.5 Batching multiple constraints 83

5.5.1 Round-by-round soundness 84
5.6 Example application of WHIR 87

6 Conclusion **92**

References **93**

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) are succinct non-interactive cryptographic proof systems that allow one party (the prover) to convince the other party (the verifier) that a given computational statement is true. Succinctness means that the proof size is sublinear with respect to the statement size and that verification is very fast compared to the computation the prover carries out. The prover may have access to private information (the witness) and if carrying out the scheme does not reveal anything about the witness to the verifier, we call it a zero knowledge SNARK (zk-SNARK). In this thesis, we do not discuss the zero-knowledge property, instead focusing on other properties of the zk-SNARK like succinctness, completeness and soundness.

Zk-SNARKs have found widespread use in privacy-preserving and verifiable blockchain systems such as Zcash [SCGGMTV14] and Ethereum [But+14], have also been used in E-voting [CGS97], designing secure and auditable databases [BBF19], privacy-preserving machine learning [MZ17] and more.

A common paradigm for getting zk-SNARKs starts by choosing a polynomial interactive oracle proof (PIOP). The PIOP is then compiled into a zk-SNARK using a polynomial commitment scheme (PCS) and the Fiat–Shamir transformation [FS86]. Such compilers rely on a random oracle, non-falsifiable assumptions or both.

Many widely used polynomial commitment schemes rely on cryptographic hardness assumptions that are not post-quantum secure. Examples include schemes based on group pairings and trusted setup, such as the KZG commitment scheme [KZG10], as well as schemes based on discrete logarithm assumptions, such as inner-product-based constructions used in Bulletproofs [BBBPWM18]. All such assumptions can be broken using a quantum computer [Sho94].

Pairing-based PCSs typically achieve very fast verification at the cost of higher prover complexity.

There is another common method of designing zk-SNARKs, introduced in [BCS16], which we call the BCS transformation. It starts with an interactive oracle proof (IOP) instead of a PIOP and transforms it into a zk-SNARK. However, nowadays the IOP itself is usually still compiled from a PIOP using an interactive oracle proof of proximity (IOPP). The composite scheme consists of the following steps:

1. A PIOP is chosen.
2. Proximity testing (an IOPP) is used to compile the PIOP into an IOP.
3. Merkle commitments are used to compile the IOP into an interactive proof (IP).
4. Finally, the IP is made non-interactive using the Fiat–Shamir transformation.

The combination of an IOPP with Merkle commitments can be viewed as a form of polynomial commitment scheme, and this method can therefore be seen as a special case of PCS-based zk-SNARK constructions. We also note that the PCS is effectively determined by the choice of the IOPP.

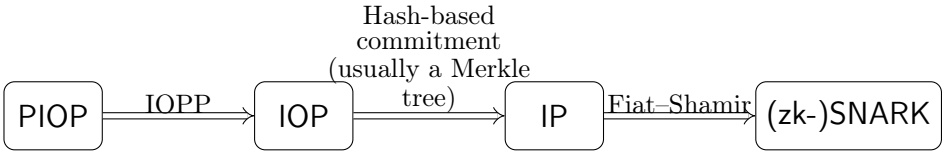

Figure 1: Constructing hash-based SNARKs from PIOPs.

This method yields a so-called *hash-based* zk-SNARK. The advantages of hash-based zk-SNARKs are that they rely solely on a random oracle and have transparent setup, they are post-quantum secure (in the quantum random oracle model) and can use small fields to get fast provers. However, these zk-SNARKs generally have slower verifiers.

Examples of IOPPs include FRI [BBHR18], BaseFold [ZCF24], Ligero [AHIV17], Brakedown (Shockwave) [GLSTW21], Orion [XZS22], STIR [ACFY24], **WHIR** [ACFY25].

The crucial step in constructing a hash-based zk-SNARK is choosing the IOPP, as different IOPPs can vary significantly in prover complexity, verifier complexity, and query complexity. This makes IOPPs an active area of cryptographic research.

The first efficient IOPP for Reed–Solomon codes was FRI [BBHR18], which introduced recursive domain folding to achieve logarithmic query complexity. Later, STIR [ACFY24] refined this approach by introducing an additional shifting step, substantially reducing the number of verifier queries required to achieve a given soundness level. In parallel, similar ideas were developed in the setting of multilinear polynomials, notably in BaseFold [ZCF24]. WHIR [ACFY25] can be seen

as combining the main ideas of STIR with a multilinear polynomial framework, yielding further improvements in verifier efficiency.

Currently, WHIR is among the most efficient known IOPPs, largely due to its very low verifier complexity. However, as it is a recent construction, little detailed analysis exists beyond the original work [ACFY25].

The goal of this thesis is to present WHIR in a self-contained and detailed manner, and to provide a careful verification of its correctness and parameter analysis. In doing so, we re-derive the main technical arguments of [ACFY25] and identify several nontrivial issues in the original presentation, including inconsistencies in indexing and quantification, as well as a flaw in a proof of round-by-round soundness (Theorem 5.7) which we correct, thereby also improving the error bound.

We start in Chapter 2 by introducing notation used throughout this work. We also introduce multilinear polynomials and the crucial Schwartz–Zippel lemma.

In Chapter 3, we start with an overview of Reed–Solomon (RS) codes and constrained RS codes, as well as list decoding. We prove the Johnson bound to illustrate the sizes of sets one can get in list decoding. We then follow [ACFY25] in defining and proving properties of important notions like mutual correlated agreement, folding, and block relative distance. In particular, we prove that up to the unique decoding radius, RS codes have mutual correlated agreement. We state a conjecture on mutual correlated agreement of RS codes up to the Johnson bound.

In Chapter 4, we define IOPs, IOPPs, and PIOPs, as well as relevant security notions like completeness and round-by-round soundness. We proceed by defining the other important ingredients in the BCS transformation: Merkle trees and the Fiat–Shamir transformation. We then show how RS code based IOPPs like WHIR can be used to convert PIOPs into IOPs. We finish the chapter by giving a brief overview of the sumcheck protocol which is an integral component of all schemes based on multilinear polynomials, including WHIR.

We start Chapter 5 by giving a high-level overview of WHIR, then formally construct WHIR, also adding diagrams to illustrate the protocol. We calculate its asymptotic parameters in detail, including deriving verifier work which is notably absent in [ACFY25]. We prove its completeness and round-by-round soundness assuming mutual correlated agreement for the underlying RS code. Round-by-round soundness is a property which is required of an IOPP to make the hash-based zk-

SNARK obtained via proximity testing and the BCS transformation secure. We finish off the chapter by giving a worked-through example of converting a PIOP into an IOP using WHIR.

2 Notation and preliminaries

We introduce some of the notation used throughout this work.

- $[n] = \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$.
- $|S|$ denotes the cardinality of the set S. $|\boldsymbol{v}|$ also denotes the length of the vector $\boldsymbol{v}$.
- If $\boldsymbol{v} = (v_1, \ldots, v_n)$ and $\boldsymbol{w} = (w_1, \ldots, w_m)$ are vectors, their **concatenation** is
$$\boldsymbol{v} \| \boldsymbol{w} = (v_1, \ldots, v_n, w_1, \ldots, w_m).$$
- $\mathbb{F}$ always denotes a finite field with characteristic different to 2. $\mathbb{F}_q$ denotes the finite field with q elements. $\mathbb{F}_q^n$ is the corresponding canonical n-dimensional vector space.
- $\mathbb{F}^*$ denotes the multiplicative group of the field $\mathbb{F}$.
- $\mathbb{F}^{<n}[X]$ denotes the set of univariate polynomials in X of degree smaller than n over the field $\mathbb{F}$. $\mathbb{F}^{<n}[X_1, \ldots, X_m]$ denotes the set of multivariate polynomials which are of degree smaller than n in each individual variable.
- The "hat" symbol over a function means that it is a low degree polynomial, i.e. $\hat{f} \in \mathbb{F}^{<d}[X]$ for a fixed degree d. We argue after Definition 2.1 that multilinear polynomials of m variables naturally correspond to univariate polynomials of degree less than 2^m. Because of this, $\hat{f}$ is also used to denote a multilinear polynomial of m variables for a fixed number m, as well as the multilinear extension (cf. Definition 2.5).
- A vector of length n over the field $\mathbb{F}$ can be viewed as a function $[n] \to \mathbb{F}$ in the obvious way. Hence, we will often use the notation $f(q)$ to mean the q-th coordinate of the vector f.
- We define $\mathrm{pow}(X, m) := \left(X^{2^0}, \ldots, X^{2^{m-1}}\right)$.
- $\alpha \leftarrow\!\!\$\ S$ means that α is sampled uniformly randomly from the set S.

- $\Pr_{\alpha \leftarrow \$ S}(X(\alpha) = 1)$ denotes the probability of the event X which depends on α when quantified over all $\alpha \in S$. In other words,

$$\Pr_{\alpha \leftarrow \$ S}(X(\alpha) = 1) = \frac{|\{\alpha \in S \mid X(\alpha) = 1\}|}{|S|}.$$

  If α is not randomly sampled from a set but obtained from some function $\texttt{fun}\colon B \to A$ with a random input, we write

$$\Pr_{\alpha \leftarrow \texttt{fun}}(X(\alpha) = 1) := \Pr_{\beta \leftarrow \$ B}(X(\texttt{fun}(\beta)) = 1) = \frac{|\{\beta \in B \mid X(\texttt{fun}(\beta)) = 1\}|}{|B|}.$$

  We sometimes use square brackets $[\cdot]$ instead of round brackets $(\cdot)$ for probabilities.

  We write w.h.p. as a shorthand for "with high probability".

- If $\mathcal{X}$ is a party in a protocol, $\mathcal{X}^y$ means that $\mathcal{X}$ has oracle access to y, that is, $\mathcal{X}$ does not know y explicitly but may query a black box (the "oracle") that returns information about y. The oracle can be viewed as a committed value accessible only through queries.

- $\mathbb{B} = \{0, 1\}$. $\mathbb{B}^m$ is called the m-dimensional **boolean hypercube**.

- If $\mathcal{L} \subseteq \mathbb{F}$ is a set, we define $\mathcal{L}^{(k)} := \{y \in \mathcal{L} : \exists x \in \mathcal{L} \text{ s.t. } x^k = y\}$, i.e. $\mathcal{L}^{(k)}$ consists of the k-th powers of elements in $\mathcal{L}$.

2.1 Multilinear polynomials

Definition 2.1 (Multilinear polynomials). *Let $\mathbb{F}$ be a field. A multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ is called **multilinear** if it is linear in each variable, i.e. the degree of the polynomial in each variable is at most 1. We write $f \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$.*

Let $f \in \mathbb{F}^{<2^m}[X]$ be a univariate polynomial. By writing the exponents of x in binary, we can define the multilinear polynomial $g \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$ extending f:

$$f(X) = g(X_1, \ldots, X_m)$$

with $f(X) = g(\mathrm{pow}(X, m))$.

Example 2.2. *Let* $\mathbb{F} = \mathbb{F}_3$ *and* $f(X) = X^7 + X^5 + 2X^4 + X^3 + X + 2$. *We can write*

$$f(X) = X^4X^2X + X^4X + 2X^4 + X^2X + X + 2 = g(X, X^2, X^4)$$

with $g(X_1, X_2, X_3) = X_1X_2X_3 + X_1X_3 + 2X_3 + X_1X_2 + X_1 + 2$.

Definition 2.3. *We define the* ***equality polynomial*** $\mathtt{eq}$ *as follows:*

$$\mathtt{eq}((X_1, \ldots, X_m), (Y_1, \ldots, Y_m)) = \prod_{i=1}^{m} \Big(X_iY_i + (1 - X_i)(1 - Y_i) \Big).$$

Notice that $\mathtt{eq}$ is clearly multilinear and extends the equality predicate on the Boolean hypercube in the sense that if

$$(x_1, \ldots, x_m, y_1, \ldots, y_m) \in \mathbb{B}^{2m},$$

then

$$\mathtt{eq}((x_1, \ldots, x_m), (y_1, \ldots, y_m)) = \begin{cases} 1 & \text{if } (x_1 = y_1) \,\&\, \ldots \,\&\, (x_m = y_m), \\ 0 & \text{else.} \end{cases}$$

The next proposition shows that the polynomials $\mathtt{eq}(\cdot, \boldsymbol{b})$, $\boldsymbol{b} \in \mathbb{B}^m$, effectively function as Lagrange basis polynomials for multilinear functions.

Proposition 2.4. *Let* $f \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$ *and* $\boldsymbol{z} \in \mathbb{F}^m$. *Then*

$$f(\boldsymbol{z}) = \sum_{\boldsymbol{b} \in \mathbb{B}^m} f(\boldsymbol{b})\, \mathtt{eq}(\boldsymbol{z}, \boldsymbol{b}).$$

Proof. A polynomial in $\mathbb{F}^{<2}[X_1, \ldots, X_m]$ has 2^m coefficients so 2^m linearly independent points are required to interpolate it.

Let's consider the polynomial $g(X_1, \ldots, X_m) := \sum_{\boldsymbol{b} \in \mathbb{B}^m} f(\boldsymbol{b})\, \mathtt{eq}((X_1, \ldots, X_m), \boldsymbol{b})$. We have $g(\boldsymbol{b}) = f(\boldsymbol{b})$ for all $\boldsymbol{b} \in \mathbb{B}^m$. The Boolean hypercube $\mathbb{B}^m$ has 2^m elements so we have $f = g$ by the above consideration. □

The previous proposition motivates the following definition.

Definition 2.5 (Multilinear extension). *Let* $f \colon \mathbb{B}^m \to \mathbb{F}$. *Then we call* $\hat{f} \in$

$\mathbb{F}^{<2}[X_1, \ldots, X_m]$, $\hat{f} := \sum_{\boldsymbol{b} \in \mathbb{B}^m} f(\boldsymbol{b})\, \mathsf{eq}((X_1, \ldots, X_m), \boldsymbol{b})$, *the* ***multilinear extension*** *of* f.

2.2 Schwartz–Zippel

The Schwartz–Zippel lemma and its corollary are important tools for proving the soundness of WHIR. They both follow trivially from the observation that low degree polynomials can only have a small number of roots.

Lemma 2.6 (Schwartz-Zippel lemma). *Let* $f(X) \in \mathbb{F}^{<n+1}[X]$ *be a non-zero polynomial. Then*

$$\Pr_{r \leftarrow \$\mathbb{F}}(f(r) = 0) \leq \frac{n}{|\mathbb{F}|}.$$

Proof. A polynomial of degree at most n has at most n roots. □

Corollary 2.7. *Let* $f, g \in \mathbb{F}^{<n+1}[X]$ *and* $f \neq g$. *Then*

$$\Pr_{r \leftarrow \$\mathbb{F}}(f(r) = g(r)) \leq \frac{n}{|\mathbb{F}|}.$$

Proof. $f(r) = g(r)$ is equivalent to $(f-g)(r) = 0$ and $f-g$ is a non-zero polynomial of degree at most n. □

3 Reed–Solomon Codes

3.1 Error-correcting codes

In this chapter, vectors are written in bold. The i-th coordinate of the vector $\boldsymbol{v}$ is written v_i.

When information is transported over a noisy communication channel, parts of it may become corrupted. Error-correcting codes mitigate this by adding redundancy into the original message, allowing the recovery of the correct data even from a partially corrupted message. Error-correcting codes are ubiquitous in modern communication but have crucial applications in cryptography as well. In practice, only linear error-correcting codes are used. We provide a brief overview here; for a more detailed treatment, see [GRS25].

Definition 3.1. *Let $\mathbb{F}_q$ be a finite field. If $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$ over $\mathbb{F}_q$ (which we denote by $\mathcal{C} \leq \mathbb{F}_q^n$), it is called an $[n,k]_q$ **linear code**. We call n the **length** and k the **dimension** or **rank** of the code. We call $\rho = \frac{k}{n}$ the **rate** of the code. An element of $\mathcal{C}$ is called a **code word** of the code $\mathcal{C}$.*

Definition 3.2. *Let $\mathcal{C}$ be an $[n,k]_q$ code and $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{C}$ be code words of $\mathcal{C}$. The **Hamming distance** between $\boldsymbol{v}$ and $\boldsymbol{w}$ is defined as $\mathsf{d}(\boldsymbol{v}, \boldsymbol{w}) = |\{i \in [n] \mid v_i \neq w_i\}|$. The **relative Hamming distance** between $\boldsymbol{v}$ and $\boldsymbol{w}$ is defined as $\Delta(\boldsymbol{v}, \boldsymbol{w}) = \frac{\mathsf{d}(\boldsymbol{v}, \boldsymbol{w})}{n}$. If $S \subseteq \mathcal{C}$, then $\Delta(\boldsymbol{v}, S) := \min_{\boldsymbol{w} \in S} \Delta(\boldsymbol{v}, \boldsymbol{w})$.*

Remark 3.3. *The (relative) Hamming distance is a distance, i.e. it satisfies the axioms of identity, symmetry and the triangle inequality.*

Definition 3.4. *The **minimum distance** of an $[n,k]_q$ linear code $\mathcal{C}$ is $\mathsf{d} := \min\{\mathsf{d}(\boldsymbol{v}, \boldsymbol{w}) \mid \boldsymbol{v}, \boldsymbol{w} \in \mathcal{C}, \boldsymbol{v} \neq \boldsymbol{w}\}$. If the minimum distance of $\mathcal{C}$ is known, we call $\mathcal{C}$ an $[n,k,\mathsf{d}]_q$ code.*

Definition 3.5. *Let $\mathcal{C}$ be an $[n,k]_q$ linear code and $\boldsymbol{v} \in \mathcal{C}$ be a code word. The **weight** of $\boldsymbol{v}$ is defined as $\mathsf{w}(\boldsymbol{v}) = |\{i \in [n] \mid v_i \neq 0\}|$ and the **relative weight** of the code word is $\mu(\boldsymbol{v}) = \frac{\mathsf{w}(\boldsymbol{v})}{n}$.*

*The **minimum weight** $\mathsf{w}_{\min}$ of the code $\mathcal{C}$ is the minimum weight of any non-zero code word of $\mathcal{C}$, i.e. $\mathsf{w}_{\min} = \min\{\mathsf{d}(\boldsymbol{v}, \boldsymbol{0}) \mid \boldsymbol{v} \in \mathcal{C}, \boldsymbol{v} \neq \boldsymbol{0}\}$. The **relative minimum weight** of $\mathcal{C}$ is $\mu := \frac{\mathsf{w}_{\min}}{n}$.*

Remark 3.6. *The minimum distance* d *of an* $[n, k, \mathsf{d}]_q$ *linear code* $\mathcal{C}$ *is equal to its minimum weight* $\mathsf{w}_{\min}$*. Thus,* $\mu = \dfrac{\mathsf{d}}{n}$.

There is a very famous bound on the dimension of a linear code.

Proposition 3.7 (Singleton bound). *Let* $\mathcal{C}$ *be an* $[n, k, \mathsf{d}]_q$ *linear code. Then* $k \leq n - \mathsf{d} + 1$.

Proof. We delete the first $\mathsf{d} - 1$ coordinates of every code word of $\mathcal{C}$. Since the distance of $\mathcal{C}$ is d, all the resulting code words will be different. We'll be left with up to $q^{n-\mathsf{d}+1}$ code words so $q^k \leq q^{n-\mathsf{d}+1}$ and $k \leq n - \mathsf{d} + 1$. □

Definition 3.8. *If* $\mathcal{C}$ *is an* $[n, k, \mathsf{d}]_q$ *linear code with* $k = n - \mathsf{d} + 1$*, we call* $\mathcal{C}$ *a maximum distance separable code or an* ***MDS code****.*

Definition 3.9. *Let* $\mathcal{C}$ *be an* $[n, k, \mathsf{d}]_q$ *linear code,* $\boldsymbol{w} \in \mathbb{F}_q^n$ *and* $\delta \in [0, 1]$*. We denote* $\mathrm{List}(\mathcal{C}, \boldsymbol{w}, \delta) := \{\boldsymbol{c} \in \mathcal{C} \mid \Delta(\boldsymbol{c}, \boldsymbol{w}) \leq \delta\}$*. We say that* $\mathcal{C}$ *is* (δ, ℓ)***-list decodable*** *if* $|\mathrm{List}(\mathcal{C}, \boldsymbol{w}, \delta)| \leq \ell$ *for all* $\boldsymbol{w} \in \mathbb{F}_q^n$.

So $|\mathrm{List}(\mathcal{C}, \boldsymbol{w}, \delta)|$ is the number of code words of $\mathcal{C}$ which are closer than δ to $\boldsymbol{w}$.

The minimum distance of a code gives an indication of how "far" code words are from each other. We often model the corruption in a channel by the addition of a random (low-weight) vector $\mathbf{e} \in \mathbb{F}_q^n$ to the code word $\mathbf{c} \in \mathcal{C}$, i.e., the received vector is $\mathbf{c} + \mathbf{e}$. Now, if the relative minimum distance of $\mathcal{C}$ is μ and $\mu(\mathbf{e}) < \dfrac{\mu}{2}$, we can correct the error in principle by decoding $\mathbf{v}+\mathbf{e}$ to the nearest code word in terms of relative Hamming distance. We thus call $\dfrac{\mu}{2}$ the **unique decoding distance** of $\mathcal{C}$.

We therefore have the implication

$$\delta \leq \frac{\mu}{2} \implies \mathrm{List}(\mathcal{C}, \boldsymbol{w}, \delta) = 1.$$

We now prove a very important bound on the size of the list given δ and μ. While we do not directly rely on it in this thesis, it illustrates the sizes of list-decoding. Admitting Conjecture 4.12 from [ACFY25] about mutual correlated agreement (cf. Chapter 3.4) up to the Johnson bound allows for a speedup of the WHIR protocol by choosing a larger δ while retaining round-by-round soundness.

The proof is based on the proof for the binary field case from [GRS25].

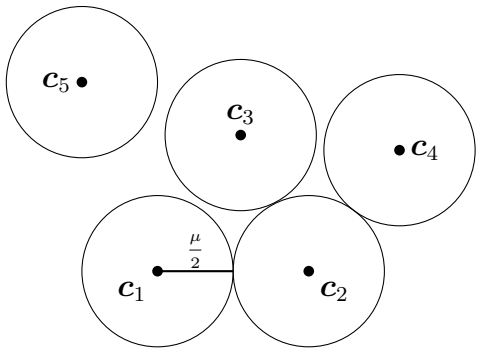

Figure 2: The unique decoding distance: no circle of radius $\dfrac{\mu}{2}$ around a code word intersects another such circle.

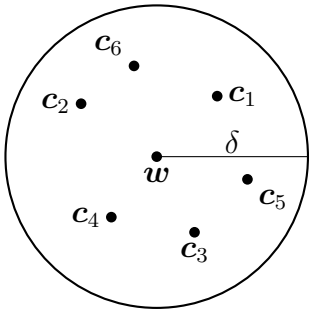

Figure 3: List decoding: the ball of radius δ around $\boldsymbol{w}$ contains up to $\ell = 6$ codewords.

Theorem 3.10 (Johnson bound). *Let $\mathcal{C}$ be an $[n, k, \mathsf{d}]_q$ linear code, $\boldsymbol{w} \in \mathbb{F}_q^n$, and*

$$0 < \delta < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right).$$

Then

$$|\operatorname{List}(\mathcal{C}, \boldsymbol{w}, \delta)| \leq \frac{\mu}{\frac{q}{q-1}\delta^2 - 2\delta + \mu}.$$

Proof. Let $\mathcal{C}$ be an $[n, k, \mathsf{d}]_q$ linear code, $\boldsymbol{w} \in \mathbb{F}_q^n$ and $\delta < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right)$. Let $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M$ be the code words of $\mathcal{C}$ with $\Delta(\boldsymbol{c}_i, \boldsymbol{w}) < \delta$. We want to show that $M \leq \frac{\mu}{\frac{q}{q-1}\delta^2 - 2\delta + \mu}$.

Let $\boldsymbol{c}_i' := \boldsymbol{c}_i - \boldsymbol{w}$, $i \in [M]$. We define

$$S = \sum_{1 \leq i < j \leq M} \Delta(\mathbf{c}_i', \mathbf{c}_j').$$

We have

$$S \geq \frac{M(M-1)}{2}\mu \tag{1}$$

because $\Delta(\mathbf{c}_i', \mathbf{c}_j') = \Delta(\mathbf{c}_i, \mathbf{c}_j) \geq \mu$ if $i \neq j$.

By choice of the $\mathbf{c}_i$, we also have

$$\mu(\mathbf{c}_i') \leq \delta, \quad i \in [M]. \tag{2}$$

Let us consider the $n \times M$ matrix whose columns are the $\mathbf{c}_i'$. Let m_i^j denote the number of columns whose i-th coordinate equals $j \in \mathbb{F}_q$. Fix a row index $i \in [n]$. For each symbol $j \in \mathbb{F}_q$, there are m_i^j columns in which the i-th coordinate equals j. Each pair of columns with different symbols in row i contributes $1/n$ to S, and there are $\sum_{j=0}^{q-1} m_i^j(M - m_i^j)$ such pairs. Hence the i-th row contributes $\frac{1}{2n}\sum_{j=0}^{q-1} m_i^j(M - m_i^j)$ to S and thus,

$$S = \sum_{i=1}^{n} \frac{1}{2n} \sum_{j=0}^{q-1} m_i^j(M - m_i^j).$$

Let $\bar{d}$ be such that $\bar{d}M = \sum_{i=1}^{n}\sum_{j=1}^{q-1} m_i^j = \sum_{i=1}^{n}(M - m_i^0)$ and let $\bar{\delta} = \bar{d}/n$. Note that by the definition of $\mu(\cdot)$ and (2)

$$\sum_{i=1}^{n}(M - m_i^0) = n\sum_{j=1}^{M}\mu(\mathbf{c}'_j) \leq nM\delta.$$

Dividing the inequality by nM, we get

$$\bar{\delta} \leq \delta. \tag{3}$$

Using the definition of $\bar{\delta}$ and the identity $x(y-x) \equiv -(y-x)^2 + y(y-x)$:

$$\begin{aligned}
S =& \frac{1}{2n}\sum_{j=0}^{q-1}\sum_{i=1}^{n} m_i^j(M - m_i^j) \\
=& \frac{1}{2n}\sum_{i=1}^{n} m_i^0(M - m_i^0) + \frac{1}{2n}\sum_{j=1}^{q-1}\sum_{i=1}^{n} m_i^j(M - m_i^j) \\
=& -\frac{1}{2n}\sum_{i=1}^{n}(M - m_i^0)^2 + \frac{1}{2n}\sum_{i=1}^{n} M(M - m_i^0) \\
&+ \frac{M}{2n}\sum_{i=1}^{n}\sum_{j=1}^{q-1} m_i^j - \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{q-1}\left(m_i^j\right)^2 \\
=& -\frac{1}{2n}\sum_{i=1}^{n}(M - m_i^0)^2 + \frac{\bar{\delta}M^2}{2} + \frac{\bar{\delta}M^2}{2} - \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{q-1}\left(m_i^j\right)^2 \\
=& \bar{\delta}M^2 - \frac{1}{2n}\sum_{i=1}^{n}(M - m_i^0)^2 - \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{q-1}\left(m_i^j\right)^2.
\end{aligned}$$

By the Cauchy-Schwarz inequality,

$$\frac{\sum_{i=1}^{n}(M - m_i^0)^2}{n} \geq \left(\sum_{i=1}^{n}\frac{M - m_i^0}{n}\right)^2 = \frac{\bar{d}^2M^2}{n^2} = \bar{\delta}^2M^2$$

and

$$\begin{aligned}\frac{\sum_{i=1}^{n}\frac{\sum_{j=1}^{q-1}\left(m_i^j\right)^2}{q-1}}{n} &\geq \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{q-1}\frac{m_i^j}{q-1}\right)^2}{n} \\ &\geq \left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{q-1}m_i^j}{n(q-1)}\right)^2 \\ &= \frac{\bar{d}^2M^2}{n^2(q-1)^2} = \frac{\bar{\delta}^2M^2}{(q-1)^2}.\end{aligned}$$

Substituting these results, we get

$$\begin{aligned}S &= \bar{\delta}M^2 - \frac{1}{2n}\sum_{i=1}^{n}(M-m_i^0)^2 - \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{q-1}\left(m_i^j\right)^2 \\ &\leq \bar{\delta}M^2 - \frac{\bar{\delta}^2M^2}{2} - \frac{\bar{\delta}^2M^2}{2(q-1)} \\ &= M^2\left(\bar{\delta} - \frac{\bar{\delta}^2}{2} - \frac{\bar{\delta}^2}{2(q-1)}\right) \\ &= M^2\left(\bar{\delta} - \frac{q}{q-1}\cdot\frac{\bar{\delta}^2}{2}\right).\end{aligned}$$

By (1),

$$\frac{M(M-1)}{2}\mu \leq M^2\left(\bar{\delta} - \frac{q}{q-1}\cdot\frac{\bar{\delta}^2}{2}\right).$$

which implies that (as $M > 0$)

$$M\left(\frac{\mu}{2} - \bar{\delta} + \frac{q}{q-1}\cdot\frac{\bar{\delta}^2}{2}\right) \leq \frac{\mu}{2}.$$

Completing the square, we get

$$M\left(\frac{q}{2(q-1)}\left(\bar{\delta} - \frac{q-1}{q}\right)^2 + \frac{\mu}{2} - \frac{q-1}{2q}\right) \leq \frac{\mu}{2}.$$

Under our assumption that $\delta < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right)$, the expression inside

the brackets is positive and hence

$$M \leq \frac{\mu}{\frac{q}{q-1}\left(\bar{\delta} - \frac{q-1}{q}\right)^2 + \mu - \frac{q-1}{q}} \leq \frac{\mu}{\frac{q}{q-1}\left(\delta - \frac{q-1}{q}\right)^2 + \mu - \frac{q-1}{q}} = \frac{\mu}{\frac{q}{q-1}\delta^2 - 2\delta + \mu}$$

where the second inequality follows from (3): $0 < \bar{\delta} \leq \delta < \dfrac{q-1}{q}$. □

In many cryptographic applications q is taken to be very large. This motivates us to simplify the bound.

Corollary 3.11 (Simplified Johnson bound). *Let $\mathcal{C}$ be an $[n, k, \mathsf{d}]_q$ linear code. For any δ satisfying*

$$0 < \delta < 1 - \sqrt{1-\mu},$$

we have

$$|\operatorname{List}(\mathcal{C}, \boldsymbol{w}, \delta)| \leq \frac{1}{\epsilon_\delta}, \qquad \epsilon_\delta = 2\sqrt{1-\mu}\big(1 - \sqrt{1-\mu} - \delta\big).$$

Proof. The Johnson bound on δ is

$$\delta < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right)$$

according to Theorem 3.10. The function

$$\frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right)$$

is monotonically decreasing in q and converges to $1-\sqrt{1-\mu}$. Hence $\delta < 1-\sqrt{1-\mu}$ implies the Johnson bound condition

$$\delta < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right).$$

Secondly, the bound on the list size can be simplified:

$$\begin{aligned}
\frac{\mu}{\frac{q}{q-1}\delta^2 - 2\delta + \mu} &= \frac{\mu}{\left(1 + \frac{1}{q-1}\right)\delta^2 - 2\delta + \mu} \\
&= \frac{\mu}{\delta^2 - 2\delta + \mu + \frac{\delta^2}{q-1}} \\
&= \frac{\mu}{(1-\delta)^2 - \left(\sqrt{1-\mu}\right)^2 + \frac{\delta^2}{q-1}} \\
&= \frac{\mu}{\left(1 - \delta + \sqrt{1-\mu}\right)\left(1 - \delta - \sqrt{1-\mu}\right) + \frac{\delta^2}{q-1}} \\
&\leq \frac{\mu}{\left(1 - \delta + \sqrt{1-\mu}\right)\left(1 - \delta - \sqrt{1-\mu}\right)} \\
&\leq \frac{\mu}{2\sqrt{1-\mu}\left(1 - \sqrt{1-\mu} - \delta\right)} \\
&\leq \frac{1}{2\sqrt{1-\mu}\left(1 - \sqrt{1-\mu} - \delta\right)} \\
&= \frac{1}{\epsilon_\delta}
\end{aligned}$$

□

Remark 3.12. *As δ approaches the Johnson radius*

$$\frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\mu}\right),$$

the bound on the list size diverges. Thus the Johnson bound is meaningful only when δ is bounded away from this radius. The same qualitative behavior is reflected by the simplified bound above.

It will come in useful to define some notions of closeness.

Definition 3.13. *Let $\boldsymbol{w} \in \mathbb{F}_q^n$ and $\delta \in [0,1]$. We say that $\boldsymbol{w}$ is δ-**close** to a code $\mathcal{C} \leq \mathbb{F}_q^n$ if there exists some $\boldsymbol{c} \in \mathcal{C}$ such that $\Delta(\boldsymbol{c}, \boldsymbol{w}) \leq \delta$ (i.e. $\mathrm{List}(\mathcal{C}, \boldsymbol{w}, \delta) \geq 1$). We denote $\Delta(\boldsymbol{w}, \mathcal{C}) \leq \delta$. If this is not the case, we say that $\boldsymbol{w}$ is δ-**far** from $\mathcal{C}$ and write $\Delta(\boldsymbol{w}, \mathcal{C}) > \delta$.*

Finally, we define the interleaved code of $\mathcal{C}$ which is obtained by writing code words of $\mathcal{C}$ next to each other.

Definition 3.14. *Let $\mathcal{C} \leq \mathbb{F}^n$ be a code. We call $\mathcal{C}^\ell \leq (\mathbb{F}^\ell)^n$ the* ***ℓ-interleaved code****. To get a code word of $\mathcal{C}^\ell$, we write ℓ code words of $\mathcal{C}$ as rows of an ℓ-by-n matrix. We collapse each of the columns into a single symbol of $\mathbb{F}^\ell$, the resulting* 1*-by-n matrix is our code word.*

Example 3.15. *Let $\mathcal{C} \leq \mathbb{F}_2^3$ be $\mathcal{C} = \{000, 010, 101, 111\}$. The corresponding* 2*-interleaved code is*

$$\begin{aligned}\mathcal{C}^2 = \{&(00,00,00),(00,01,00),(01,00,01),(01,01,01),\\ &(00,10,00),(00,11,00),(01,10,01),(01,11,01),\\ &(10,00,10),(10,01,10),(11,00,11),(11,01,11),\\ &(10,10,10),(10,11,10),(11,10,11),(11,11,11)\}.\end{aligned}$$

The Hamming distance between code words is calculated with respect to the alphabet $\mathbb{F}^2$. So $d((10,00,10),(01,00,01)) = 2$, for example.

3.2 Reed–Solomon Codes

Let $\mathcal{L} \subseteq \mathbb{F}$, we call $\mathcal{L}$ the **evaluation domain**. We always assume that $\mathcal{L}$ has a known fixed ordering.

Definition 3.16. *The* ***Reed–Solomon (RS) code*** *of degree d over the field $\mathbb{F}$ with evaluation domain $\mathcal{L}$ is the linear code whose code words are evaluations over $\mathcal{L}$ of univariate polynomials of degree (strictly) less than d.*

In this work we only consider RS codes with **smooth** evaluation domains. If an RS code has a smooth evaluation domain, we sometimes call the code itself smooth. In the case of smooth RS codes we can use multilinear polynomials to give an alternative description to the code words.

Definition 3.17 ([ACFY25] Definition 2). *The* $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ *code is said to have a* ***smooth evaluation domain*** *if $\mathcal{L}$ is a multiplicative coset of $\mathbb{F}^*$ with $|\mathcal{L}|$ a power of 2 and $d = 2^m$, where $m \in \mathbb{N}$.*

The Reed–Solomon code over the field $\mathbb{F}$ with smooth evaluation domain $\mathcal{L}$ is the

linear code

$$\begin{aligned}&\mathrm{RS}[\mathbb{F},\mathcal{L},m] := \{f\colon \mathcal{L}\to\mathbb{F} : \exists \hat{f}\in\mathbb{F}^{<d}[X] \text{ s.t. } \forall x\in\mathcal{L}, \hat{f}(x)=f(x)\}\\ &= \{f\colon \mathcal{L}\to\mathbb{F} : \exists \hat{f}\in\mathbb{F}^{<2}[X_1,\ldots,X_m] \text{ s.t. } \forall x\in\mathcal{L}, \hat{f}(\mathrm{pow}(x,m))=f(x)\}.\end{aligned}$$

We call m the ***number of variables****.*

Let $d < n = |\mathcal{L}|$. Since a polynomial of degree less than d is uniquely determined by its values on any d distinct points, there is a one-to-one correspondence between polynomials of degree $< d$ and code words of the $\mathrm{RS}[\mathbb{F},\mathcal{L},m]$ code. Thus, the dimension of the code is $d = 2^m$ and the rate is $\rho = \dfrac{d}{n} = \dfrac{2^m}{n}$.

Since a non-zero polynomial of degree $< d$ can have up to $d-1$ roots, and there exist polynomials that have $d-1$ different roots in $\mathcal{L}$, the minimum weight of the code is $n-d+1$. We have proved the following proposition (cf. Definition 3.8).

Proposition 3.18. *Let $|\mathcal{L}| = n$ and $d < n$. The $\mathrm{RS}[\mathbb{F},\mathcal{L},m]$ code is an MDS code.*

The unique decoding distance is

$$\frac{\mu}{2} = \frac{n-d+1}{2n} = \frac{1-\rho}{2} + \frac{1}{2n}$$

where $\rho = d/n$ denotes the code rate. We also got $\rho - \dfrac{1}{n} = 1-\mu$ which gives the following result (cf. Corollary 3.11).

Theorem 3.19 (Simplified Johnson bound for Reed–Solomon codes). *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F},\mathcal{L},m]$, $\boldsymbol{w}\colon \mathcal{L}\to\mathbb{F}$. For any δ satisfying*

$$\delta < 1-\sqrt{\rho},$$

we have

$$|\operatorname{List}(\mathcal{C},\boldsymbol{w},\delta)| \le \frac{1}{\epsilon_\delta}, \qquad \epsilon_\delta := 2\sqrt{\rho-1/n}(1-\sqrt{\rho}-\delta).$$

Proof. From Corollary 3.11, we have

$$\delta < 1-\sqrt{1-\mu} = 1-\sqrt{\rho-1/n}.$$

Since $1-\sqrt{\rho} < 1-\sqrt{\rho-1/n}$, we can take $\delta < 1-\sqrt{\rho}$ and the Johnson bound is satisfied.

The bound on the list size from Corollary 3.11 is

$$|\operatorname{List}(\mathcal{C}, \boldsymbol{w}, \delta)| \leq \frac{1}{2\sqrt{1-\mu}\big(1-\sqrt{1-\mu}-\delta\big)}.$$

We have

$$\begin{aligned}\frac{1}{2\sqrt{1-\mu}\big(1-\sqrt{1-\mu}-\delta} &= \frac{1}{2\sqrt{\rho-\frac{1}{n}}\left(1-\sqrt{\rho-\frac{1}{n}}-\delta\right)} \\ &\leq \frac{1}{2\sqrt{\rho-\frac{1}{n}}\left(1-\sqrt{\rho}-\delta\right)}.\end{aligned}$$

□

Remark 3.20. *For large n, the difference between ρ and $\rho - \dfrac{1}{n}$ is negligible. In applications it is common to ignore the additive $1/n$ term and write*

$$|\operatorname{List}(\mathcal{C}, \boldsymbol{w}, \delta)| \lesssim \frac{1}{2\sqrt{\rho}(1-\sqrt{\rho}-\delta)}.$$

WHIR uses a certain class of subcodes of smooth Reed–Solomon codes, called **constrained Reed–Solomon** codes, which consist of code words which satisfy an algebraic constraint of a particular form. The constraint is used to encode verifier's challenges directly into the code.

Definition 3.21 ([ACFY25] Definition 4.5)**.** *The **constrained Reed–Solomon code** with field $\mathbb{F}$, smooth evaluation domain $\mathcal{L} \subseteq \mathbb{F}$, number of variables $m \in \mathbb{N}$, weight polynomial $\hat{w} \in \mathbb{F}[Z, X_1, \ldots, X_m]$, and target $\sigma \in \mathbb{F}$ is*

$$\operatorname{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{w}, \sigma] := \left\{ f \in \operatorname{RS}[\mathbb{F}, \mathcal{L}, m] : \sum_{\boldsymbol{b} \in \mathbb{B}^m} \hat{w}(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sigma \right\}$$

where $\hat{f} \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$ denotes the multilinear polynomial corresponding to f.

We recall that we can evaluate $\hat{f}$ at any point $\boldsymbol{\alpha} \in \mathbb{F}^m$ with the formula

$$\hat{f}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{b} \in \mathbb{B}^m} \hat{f}(\boldsymbol{b}) \, \mathsf{eq}(\boldsymbol{\alpha}, \boldsymbol{b})$$

by virtue of Proposition 2.4. Thus, by setting

$$\hat{w}(Z, X_1, \ldots, X_m) = Z\,\mathsf{eq}(\boldsymbol{\alpha}, (X_1, \ldots, X_m)),$$

we can define a constrained RS code whose code words are exactly evaluations over $\mathcal{L}$ of the low degree polynomials which evaluate to a given value σ at a given point $\boldsymbol{\alpha} \in \mathbb{F}^m$. We also note that clearly $\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, 0, 0] = \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ so regular RS codes form a subset of constrained RS codes. This allows comparing the speed of constrained RS code based WHIR with other Reed–Solomon IOPPs like FRI.

We call the constraint $\sum_{\boldsymbol{b} \in \mathbb{B}^m} \hat{w}(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sigma$ a **sumcheck**. We analyze verifying sumchecks in Chapter 4.5.

We can also extend Definition 3.21 to include multiple constraints in the following way.

Definition 3.22 ([ACFY25] Definition 4.6)**.** *The* ***multi-constrained Reed–Solomon code*** *with field $\mathbb{F}$, smooth evaluation domain $\mathcal{L} \subseteq \mathbb{F}$, number of variables $m \in \mathbb{N}$, weight polynomials $\hat{w}_1, \ldots, \hat{w}_t \in \mathbb{F}[Z, X_1, \ldots, X_m]$, and targets $\sigma_1, \ldots, \sigma_t \in \mathbb{F}$ is*

$$\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_1, \sigma_1), \ldots, (\hat{w}_t, \sigma_t)] := \bigcap_{i \in [t]} \left\{ f \in \mathrm{RS}[\mathbb{F}, \mathcal{L}, m] : \sum_{\boldsymbol{b} \in \mathbb{B}^m} \hat{w}_i(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sigma_i \right\}.$$

3.3 Folding

An integral component of WHIR (as well as FRI and STIR before it) is *folding*. Folding allows us to reduce the effective size of the evaluation domain of a code, which in turn reduces the complexity of verification. Intuitively, it breaks code words into smaller parts which, in the case of a valid code word, both belong to a smaller code, and combines them.

Let $\mathbb{F}$ be a finite field with $\operatorname{char} \mathbb{F} \neq 2$ and $\mathcal{L} \subseteq \mathbb{F}$ be a smooth evaluation domain. In particular, $0 \notin \mathcal{L}$ and $|\mathcal{L}|$ is a power of 2.

Since $\mathcal{L}$ is smooth, we can write $\mathcal{L} = \{a, a\omega, \ldots, a\omega^{2^r-1}\}$ for some $a \in \mathbb{F}^*$ and ω a 2^r-th root of unity in $\mathbb{F}^*$. Since the order of ω is 2^r, we must have $\omega^{2^{r-1}} = -1$. This means that if $x \in \mathcal{L}$, then also $-x \in \mathcal{L}$. The same applies for $\mathcal{L}^{(2^k)} = \{a^{2^k}, a^{2^k}\omega^{2^k}, \ldots, a^{2^k}\omega^{2^{r-k}}\}$ since it is of the same form.

Definition 3.23 ([ACFY25] Definition 4.14)**.** *Let $f : \mathcal{L} \to \mathbb{F}$ be a function, and $\alpha \in \mathbb{F}$. We define* $\mathrm{Fold}(f, \alpha) : \mathcal{L}^{(2)} \to \mathbb{F}$ *as follows:*

$$\forall x \in \mathcal{L},\ \mathrm{Fold}(f, \alpha)(x^2) := \frac{f(x) + f(-x)}{2} + \alpha \cdot \frac{f(x) - f(-x)}{2x}. \tag{4}$$

For $k \leq m$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}^k$ we define $\mathrm{Fold}(f, \boldsymbol{\alpha}) : \mathcal{L}^{(2^k)} \to \mathbb{F}$ *by* $\mathrm{Fold}(f, \boldsymbol{\alpha}) := f_k$ *where f_k is defined recursively as follows: $f_0 := f$, and $f_i :=$* $\mathrm{Fold}(f_{i-1}, \alpha_i)$*. For a set $S \subseteq \mathbb{F}^{\mathcal{L}}$ we denote* $\mathrm{Fold}(S, \boldsymbol{\alpha}) := \{\mathrm{Fold}(f, \boldsymbol{\alpha}) \mid f \in S\}$.

We provide a mock example.

Example 3.24. *Let $\mathcal{L} \subseteq \mathbb{F}_{17}$ be the following smooth evaluation domain: $\mathcal{L} = \{3, 5, 6, 7, -7, -6, -5, -3\}$. Then $|\mathcal{L}| = 2^3$ so $k = 3$. We have $\mathcal{L}^{(2)} = \{2, 8, -8, -2\}$, $\mathcal{L}^{(4)} = \{4, -4\}$ and $\mathcal{L}^{(8)} = \{-1\}$.*

We define the polynomial function $\hat{f} \in \mathbb{F}_{17}[X]$ to be the polynomial X^4+X^3+8X+3. Then

x	0	1	2	3	4	5	6	7	8	-8	-7	-6	-5	-4	-3	-2	-1
$\hat{f}(x)$	3	-4	-8	-1	-2	-6	-1	-2	0	4	-1	-2	4	-7	-1	-5	-5

Let $\mathbb{F}_{17}^3 \ni \boldsymbol{\alpha} = (5, -2, -7)$. We fold $\hat{f}$ by $\boldsymbol{\alpha}$.

- $\mathrm{Fold}(f, 5)(x^2) = \dfrac{f(x) + f(-x)}{2} + 5 \cdot \dfrac{f(x) - f(-x)}{2x}$. *Evaluating it on $\mathcal{L}^{(2)}$ gets us the table*

x^2	2	8	-8	-2
$\mathrm{Fold}(f, 5)(x^2)$	6	-6	-1	3

- $$\mathrm{Fold}(f, (5, -2))(x^4) = \frac{\mathrm{Fold}(f, 5)(x^2) + \mathrm{Fold}(f, 5)(-x^2)}{2} - 2 \cdot \frac{\mathrm{Fold}(f, 5)(x^2) - \mathrm{Fold}(f, 5)(-x^2)}{2x^2}.$$

  Evaluating it on $\mathcal{L}^{(4)}$ gets us the table

x^4	4	-4
$\mathrm{Fold}(f, (5, -2))(x^4)$	3	-5

•

$$\begin{aligned}\mathrm{Fold}(f,\boldsymbol{\alpha}) &= \mathrm{Fold}(f,(5,-2,-7))(x^8)\\ &= \frac{\mathrm{Fold}(f,(5,-2))(x^4)+\mathrm{Fold}(f,(5,-2))(-x^4)}{2}\\ &\quad -7\cdot\frac{\mathrm{Fold}(f,(5,-2))(x^4)-\mathrm{Fold}(f,(5,-2))(-x^4)}{2x^4}.\end{aligned}$$

We get $\mathrm{Fold}(f,\boldsymbol{\alpha}) = -8$.

The following claim shows that the folding of a Reed–Solomon code word at any set of points results in a Reed–Solomon code word. It also gives a way to calculate the multilinear extension of the folding.

Proposition 3.25 ([ACFY25] Claim 4.15)**.** [1] *Let* $f : \mathcal{L} \to \mathbb{F}$ *be a function,* $\boldsymbol{\alpha} \in \mathbb{F}^k$ *folding randomness and let* $g := \mathrm{Fold}(f,\boldsymbol{\alpha})$*. If* $f \in \mathrm{RS}[\mathbb{F},\mathcal{L},m]$ *and* $k \le m$*, then* $g \in \mathrm{RS}[\mathbb{F},\mathcal{L}^{(2^k)},m-k]$*, and further, the multilinear extension of* g *is given by* $\hat{g}(X_{k+1},\dots,X_m) := \hat{f}(\boldsymbol{\alpha},X_{k+1},\dots,X_m)$ *where* $\hat{f}$ *is the multilinear extension of* f*.*

Proof. Consider a function $f' \in \mathrm{RS}[\mathbb{F},\mathcal{L}',m']$ and $\alpha \in \mathbb{F}$. First we show that $g' := \mathrm{Fold}(f',\alpha) \in \mathrm{RS}[\mathbb{F},\mathcal{L}'^{(2)},m'-1]$ and that $\hat{g}' := \hat{f}'(\alpha,X_2,\dots,X_{m'})$ where $\hat{f}' \in \mathbb{F}^{<2}[X_1,\dots,X_{m'}]$ is the multilinear extension of f', meaning that $f'(x) = \hat{f}'(\mathrm{pow}(x,m'))$ for every $x \in \mathcal{L}'$ (cf. Definition 3.17). Let $f'_0, f'_1 \in \mathbb{F}^{<2}[X_2,\dots,X_{m'}]$ be the polynomials such that

$$\hat{f}'(X_1,\dots,X_{m'}) = \hat{f}'_0(X_2,\dots,X_{m'}) + X_1\hat{f}'_1(X_2,\dots,X_{m'}).$$

We define $g' := f'_0 + \alpha f'_1$.

Let $x \in \mathcal{L}'$ and $z = x^2 \in \mathcal{L}'^{(2)}$, define $\boldsymbol{z} := \mathrm{pow}(z,m'-1) = \left(x^2,\dots,x^{2^{m'}}\right)$, and

[1] We correct several minor typographical errors in the original statement and proof.

observe that $\mathrm{pow}(x, m') = (x, \boldsymbol{z})$ and $\mathrm{pow}(-x, m') = (-x, \boldsymbol{z})$. Therefore

$$\begin{aligned}
g'(z) &= f'_0(z) + \alpha f'_1(z) \\
&= \frac{f'(x) + f'(-x)}{2} + \alpha \cdot \frac{f'(x) - f'(-x)}{2x} \\
&= \frac{\hat{f}'(\mathrm{pow}(x, m')) + \hat{f}'(\mathrm{pow}(-x, m'))}{2} + \alpha \cdot \frac{\hat{f}'(\mathrm{pow}(x, m')) - \hat{f}'(\mathrm{pow}(-x, m'))}{2x} \\
&= \frac{\hat{f}'(x, \boldsymbol{z}) + \hat{f}'(-x, \boldsymbol{z})}{2} + \alpha \cdot \frac{\hat{f}'(x, \boldsymbol{z}) - \hat{f}'(-x, \boldsymbol{z})}{2x} \\
&= \frac{\hat{f}'_0(\boldsymbol{z}) + x\hat{f}'_1(\boldsymbol{z}) + \hat{f}'_0(\boldsymbol{z}) - x\hat{f}'_1(\boldsymbol{z})}{2} + \alpha \cdot \frac{\hat{f}'_0(\boldsymbol{z}) + x\hat{f}'_1(\boldsymbol{z}) - \hat{f}'_0(\boldsymbol{z}) + x\hat{f}'_1(\boldsymbol{z})}{2x} \\
&= \hat{f}'_0(\boldsymbol{z}) + \alpha \hat{f}'_1(\boldsymbol{z}) = \hat{g}'(\boldsymbol{z}).
\end{aligned}$$

Therefore, $g' \in \mathrm{RS}[\mathbb{F}, \mathcal{L}'^{(2)}, m' - 1]$ and, by the definitions of $\hat{f}'_0$ and $\hat{f}'_1$, its multilinear extension equals $f'(\alpha, X_2, \ldots, X_{m'})$.

Finally, we prove the claim inductively. By assumption, $f \in \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$.

- The base case $k = 1$ follows by the previous argument, taking $f' = f$ and $g' = g$.
- The inductive hypothesis posits that, for every $\boldsymbol{\alpha}' \in \mathbb{F}^{k-1}$, we have $f_{k-1} := \mathrm{Fold}(f, \boldsymbol{\alpha}') \in \mathrm{RS}[\mathbb{F}, \mathcal{L}^{(2^{k-1})}, m - k + 1]$ and that $\hat{f}_{k-1} := \hat{f}(\boldsymbol{\alpha}', X_k, \ldots, X_m)$. Thus, invoking the argument above we conclude that for every $(\boldsymbol{\alpha}', \alpha_k) = \boldsymbol{\alpha} \in \mathbb{F}^k$ it holds that

$$\begin{aligned}
f_k := \mathrm{Fold}(f, \boldsymbol{\alpha}) &= \mathrm{Fold}(\mathrm{Fold}(f, \boldsymbol{\alpha}'), \alpha_k) \\
&= \mathrm{Fold}(f_{k-1}, \alpha_k) \in \mathrm{RS}[\mathbb{F}, \mathcal{L}^{(2^k)}, m - k]
\end{aligned}$$

  and that $\hat{f}_k(X_k, \ldots, X_m) := \hat{f}(\boldsymbol{\alpha}, X_{k+1}, \ldots, X_m)$.

□

We use this result to expand on Example 3.24.

Example 3.26. *Let $\mathbb{F}_{17} \supseteq \mathcal{L} = \{3, 5, 6, 7, -7, -6, -5, -3\}$ and let $\hat{f} \in \mathbb{F}_{17}[X]$ be the polynomial $X^4 + X^3 + 8X + 3$, evaluated over $\mathcal{L}$. We want to fold $f = \hat{f}|_{\mathcal{L}}$ by $\boldsymbol{\alpha} =$*

$(5, -2, -7)$. *We first redefine* $\hat{f}$ *as a multilinear polynomial* $\hat{f} \in \mathbb{F}_{17}^{<2}[X_1, X_2, X_3]$*:*

$$\hat{f}(X_1, X_2, X_3) = X_3 + X_1X_2 + 8X_1 + 3.$$

According to Proposition 3.25, we have

$$\mathrm{Fold}(f, \boldsymbol{\alpha}) = \hat{f}(\boldsymbol{\alpha}) = \hat{f}(5, -2, -7) = -8,$$

just as we did in Example 3.24.

3.4 Mutual correlated agreement

Folding (c.f. Definition 3.23) introduces a random linear combination of two functions. If both of these are close to code words, so is their random linear combination. However, security of folding requires the converse: It is difficult to find two functions which are not both close to code words but whose random linear combination is close to a code word w.h.p..

In fact, analysis of the round-by-round soundness of WHIR needs stronger results: *correlated agreement* postulates that w.h.p. if the random linear combination of functions is close to the code, these functions themselves are all close to the code but they all agree with the code *at the same coordinates*.

Mutual correlated agreement was first defined in [ACFY25] and takes this one step further: W.h.p., if the random linear combination of functions is close to the code, then so are all the functions themselves, but they all agree with the code at the same coordinates as the random linear combination does.

These notions make analysis of the batched code words much simpler: If we have mutual correlated agreement, knowing where there are errors in the received random linear combination allows us to know exactly where the errors in the original words were.

We start by formally defining correlated agreement.

Definition 3.27 ([ACFY25] Definition 4.5)**.** *Let* $\mathcal{C} \leq \mathbb{F}^n$ *be a linear code. We say that* `Gen` *is a* ***proximity generator*** *for* $\mathcal{C}$ *with proximity bound* `B` *and error* `err` *if*

the following holds for all vectors $f_1, \ldots, f_\ell \in \mathbb{F}^n$ *and* $\delta \in (0, 1 - \mathtt{B}(\mathcal{C}, \ell))$*: If*

$$\Pr_{\mathbb{F}^\ell \ni (r_1,\ldots,r_\ell) \leftarrow \mathtt{Gen}(\ell)} \left[\Delta\left(\sum_{i \in [\ell]} r_i f_i, \mathcal{C} \right) \leq \delta \right] > \mathtt{err}(\mathcal{C}, \ell, \delta)$$

then there exists $S \subseteq [n]$ *with* $|S| \geq (1 - \delta)n$ *and*

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

Intuitively, if the random coefficients $r_1, \ldots, r_\ell$ generated by a proximity generator $\mathtt{Gen}$ make the linear combination $\sum_{i \in [\ell]} r_i f_i$ close to a code word of $\mathcal{C}$ with non-negligible probability, then there must exist a large set S where each f_i locally agrees with some code word of $\mathcal{C}$ (but not necessarily the same one for different values of i).

Figure 4 illustrates the definition.

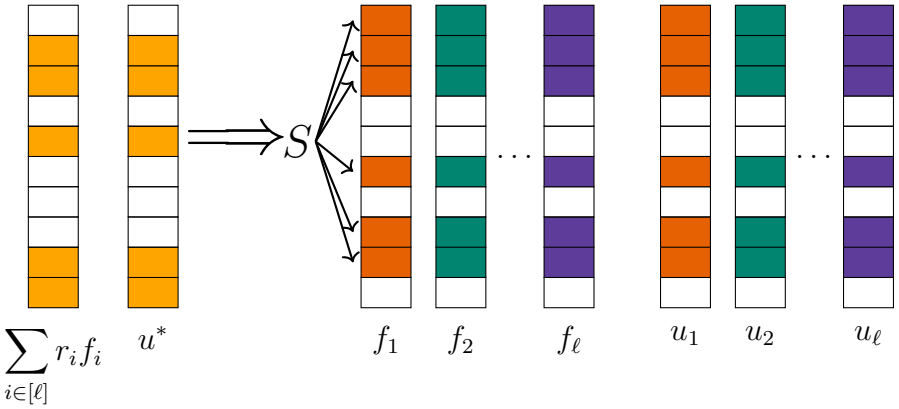

Figure 4: Correlated agreement: If $\sum_{i \in [\ell]} r_i f_i$ is close to a code word $u^* \in C$ then there exists a set S such that the f_i are close to code words $u_i \in C$, $i \in [\ell]$. Cells of the same color in the same row contain the same field element.

It was shown in [BCIKS20] that Reed–Solomon codes have good proximity generators.

Theorem 3.28 ([BCIKS20])**.** *Let* $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ *be a Reed–Solomon code with rate* $\rho = \dfrac{2^m}{|\mathcal{L}|}$*. The function* $\mathtt{Gen}(\ell; \alpha) = (1, \alpha, \ldots, \alpha^{\ell-1})$ *is a proximity generator for* $\mathcal{C}$ *with proximity bound* $\mathtt{B}(\mathcal{C}, \ell) = \sqrt{\rho}$ *and the error* $\mathtt{err}(\mathcal{C}, \ell, \delta)$ *defined below.*

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

$$\mathtt{err}(\mathcal{C}, \ell, \delta) := \frac{2^m(\ell-1)}{|\mathbb{F}|\rho}.$$

- *If* $\delta \in \left(\dfrac{1-\rho}{2}, 1-\mathtt{B}(\mathcal{C}, \ell)\right)$ *then*

$$\mathtt{err}(\mathcal{C}, \ell, \delta) := \frac{4^m(\ell-1)}{|\mathbb{F}|\left(2\min\left\{1-\sqrt{\rho}-\delta, \frac{\sqrt{\rho}}{20}\right\}\right)^7}.$$

There is need of a stronger notion. We say that a proximity generator has mutual correlated agreement if w.h.p. the existence of the set S implies that both the f_i, $i \in [\ell]$, and $\sum_{i\in[\ell]} r_i f_i$ agree with the code $\mathcal{C}$ in the same set S.

Definition 3.29 ([ACFY25] Definition 4.9)**.** *Let* $\mathcal{C} \leq \mathbb{F}^n$ *be a linear code. We say that* Gen *is a* ***proximity generator*** *for* $\mathcal{C}$ *with* ***mutual correlated agreement*** *with proximity bound* $\mathtt{B}^\star$ *and error* $\mathtt{err}^\star$ *if for all vectors* $f_1, \ldots, f_\ell \in \mathbb{F}^n$ *and* $\delta \in (0, 1-\mathtt{B}^\star(\mathcal{C}, \ell))$*, the following holds:*

$$\Pr_{(r_1,\ldots,r_\ell)\leftarrow\mathtt{Gen}(\ell)}\left[\begin{array}{c} |S| \geq (1-\delta)n, \\ \exists S \subseteq [n]\ s.t.\ \exists u \in \mathcal{C},\ \forall x \in S, u(x) = \sum_{j\in[\ell]} r_j f_j(x), \\ \exists i \in [\ell],\ \forall u_i' \in \mathcal{C}, \exists x \in S,\ u_i'(x) \neq f_i(x) \end{array}\right] \leq \mathtt{err}^\star(\mathcal{C}, \ell, \delta).$$

Remark 3.30. *Using elementary logic, we can rewrite the statement as*

$$\Pr_{(r_1,\ldots,r_\ell)\leftarrow\mathtt{Gen}(\ell)}\left[\forall S \subseteq [n], \left(\begin{array}{c} |S| \geq (1-\delta)n \\ \& \\ \exists u \in \mathcal{C},\ \forall x \in S, u(x) = \sum_{j\in[\ell]} r_j f_j(x) \\ \Downarrow \\ \forall i \in [\ell], \exists u_i' \in \mathcal{C}, \forall x \in S,\ u_i'(x) = f_i(x) \end{array}\right)\right] > \mathtt{err}^\star(\mathcal{C}, \ell, \delta).$$

We always tacitly assume that $\mathtt{B}^\star(\mathcal{C}, \ell) \geq \rho$. This implies that $\delta < 1-\rho$, i.e., δ is inside the famous capacity bound.

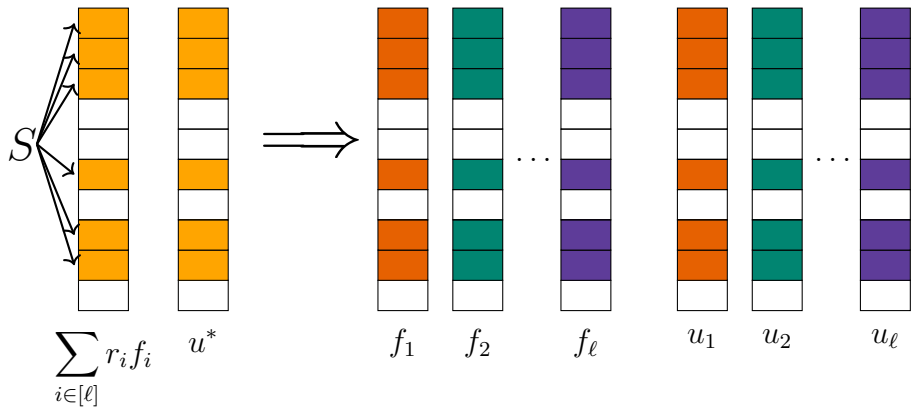

Figure 5: Mutual correlated agreement: $\sum_{i\in[\ell]} r_i f_i$ agrees with the code in the same set S as all the f_i.

We prove that in the unique decoding region, i.e. when $1-\delta > 1-\frac{\mu}{2}$, every proximity generator exhibits mutual correlated agreement. We modified the definition of $\mathtt{B}^\star$ from the one in [ACFY25] by replacing a minimum with a maximum; this change is necessary for the proof to go through.

Lemma 3.31 ([ACFY25] Lemma 4.10)**.** *Let $\mathcal{C}$ be a linear code with relative minimum distance μ and let* `Gen` *be a proximity generator for $\mathcal{C}$ with proximity bound* `B` *and error* `err`*. Then* `Gen` *has mutual correlated agreement with proximity bound* $\mathtt{B}^\star(\mathcal{C},\ell) = \max\left\{1-\frac{\mu}{2}, \mathtt{B}(\mathcal{C},\ell)\right\}$ *and error* $\mathtt{err}^\star(\mathcal{C},\ell,\delta) = \mathtt{err}(\mathcal{C},\ell,\delta)$.

Proof of Lemma 3.31. We omit arguments like $\mathcal{C},\ell,\delta$ if they're clear from the context. Fix $f_1,\ldots,f_\ell \in \mathbb{F}^n$ and $\delta \in (0, 1-\mathtt{B}^\star)$. Then $0 < \delta < \min\left\{\frac{\mu}{2}, 1-\mathtt{B}\right\}$.

Let $T \subseteq [n]$ be a maximal set such that for every f_i there exists $u_i \in \mathcal{C}$ such that for all $x \in T$, $f_i(x) = u_i(x)$. Suppose that we do not have mutual correlated agreement with error $\mathtt{err}^\star$:

$$\Pr_{(r_1,\ldots,r_\ell)\leftarrow\mathtt{Gen}(\ell)}\left[\begin{array}{c} |S| \geq (1-\delta)n, \\ \exists S\subseteq[n] \text{ s.t. } \exists u\in\mathcal{C},\ \forall x\in S, u(x) = \sum_{j\in[\ell]} r_j f_j(x), \\ \exists i\in[\ell],\ \forall u'\in\mathcal{C}, \exists x\in S,\ u'(x)\neq f_i(x) \end{array}\right] > \mathtt{err}^\star = \mathtt{err}. \tag{5}$$

We show that this results in a contradiction with the maximality of T. But every finite ordered set has a maximal element so T must exist. It follows that we must

have mutual correlated agreement.

Clearly, $\Delta\left(\sum_{i\in[\ell]} r_i f_i, \mathcal{C}\right) \leq \delta$ whenever the event above occurs. Since $\texttt{Gen}$ is a proximity generator with error $\texttt{err}$ and $\delta < 1 - \texttt{B}$, we deduce that there exists a set $A \subseteq [n]$ with $|A| \geq (1-\delta)n$ such that for every $i \in [\ell]$ there is a code word $u_i' \in \mathcal{C}$ with $\forall x \in A$, $u_i'(x) = f_i(x)$. Since T is maximal, we have: $|T| \geq |A| \geq (1-\delta)n$.

Fix $\boldsymbol{r} = (r_1, \ldots, r_\ell)$ in the image of $\texttt{Gen}(\ell)$ and define $g_{\boldsymbol{r}} := \sum_{i\in[\ell]} r_i f_i$ and $w_{\boldsymbol{r}} := \sum_{i\in[\ell]} r_i u_i$. Since $\mathcal{C}$ is linear $w_{\boldsymbol{r}} \in \mathcal{C}$. Note that

$$\forall x \in T, g_{\boldsymbol{r}}(x) = \sum_{i\in[\ell]} r_i f_i(x) = \sum_{i\in[\ell]} r_i u_i(x) = w_{\boldsymbol{r}}(x).$$

Since $|T| \geq (1-\delta)n$, we conclude that $w_{\boldsymbol{r}} \in \mathrm{List}(\mathcal{C}, g_{\boldsymbol{r}}, \delta)$.

We consider a set $S \subseteq [n]$ such that $|S| \geq (1-\delta)n$ and $g_{\boldsymbol{r}}$ agrees with $w' \in \mathcal{C}$ on S, but there exists f_i, $i \in [\ell]$, such that f_i does not agree on all of S with any code word of $\mathcal{C}$ (i.e. S is the set we get from (5)). Observe the following.

- $S \setminus T \neq \emptyset$: this follows since for every f_i, $i \in [\ell]$, there is a code word that agrees with f_i on all of T.
- $w' = w_{\boldsymbol{r}}$: this holds since $w_{\boldsymbol{r}}, w' \in \mathrm{List}(\mathcal{C}, g_{\boldsymbol{r}}, \delta)$ and we have $|\mathrm{List}(\mathcal{C}, g_{\boldsymbol{r}}, \delta)| \leq 1$ as $\delta < \frac{\mu}{2}$.

Combining these two observations we deduce that the set $S \setminus T$ is nonempty and that, for every $x \in S \setminus T$, $g_{\boldsymbol{r}}(x) = w'(x) = w_{\boldsymbol{r}}(x)$. Recalling that $\forall x \in T$, $g_{\boldsymbol{r}}(x) = w_{\boldsymbol{r}}(x)$, we conclude that

$$\Delta(g_{\boldsymbol{r}}, \mathcal{C}) \leq \Delta(g_{\boldsymbol{r}}, w_{\boldsymbol{r}}) \leq \frac{n - |S \cup T|}{n} < 1 - \frac{|T|}{n}.$$

Therefore, by (5),

$$\begin{aligned}\texttt{err} = \texttt{err}^\star &< \Pr_{(r_1,\ldots,r_\ell)\leftarrow\texttt{Gen}(\ell)}\left[\begin{array}{c} |S| \geq (1-\delta)n, \\ \exists S \subseteq [n] \text{ s.t. } \exists u \in \mathcal{C},\, \forall x \in S, u(x) = \sum_{j\in[\ell]} r_j f_j(x), \\ \exists i \in [\ell],\, \forall u' \in \mathcal{C}, \exists x \in S,\, u'(x) \neq f_i(x) \end{array}\right] \\ &\leq \Pr_{(r_1,\ldots,r_\ell)\leftarrow\texttt{Gen}(\ell)}\left[\Delta\left(\sum_{i\in[\ell]} r_i f_i, \mathcal{C}\right) < 1 - \frac{|T|}{n}\right].\end{aligned}$$

We can once again apply the fact that $\texttt{Gen}$ is a proximity generator for $\mathcal{C}$ since $1 - \frac{|T|}{n} \leq \delta < 1 - \texttt{B}$. Hence, there exists a set $W \subseteq [n]$ with $|W| > |T|$ such that for every f_i, $i \in [\ell]$ there exists $u_i \in \mathcal{C}$ such that $\forall x \in W$, $f_i(x) = u_i(x)$. This contradicts the maximality of T. □

We get the following corollary for RS codes.

Corollary 3.32 ([ACFY25] Corollary 4.11)**.** *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ be a Reed–Solomon code with rate $\rho = 2^m/|\mathcal{L}|$. The function $\texttt{Gen}(\ell;\alpha) = (1, \alpha, \ldots, \alpha^{\ell-1})$ is a proximity generator for $\mathcal{C}$ with mutual correlated agreement with proximity bound $\texttt{B}^\star(\mathcal{C},\ell) := \frac{1+\rho}{2}$ and error $\texttt{err}^\star(\mathcal{C},\ell,\delta) = \frac{2^m(\ell-1)}{|\mathbb{F}|\rho}$.*

Proof. We have

$$1 - \frac{\mu}{2} = \frac{1+\rho}{2} - \frac{1}{2n}$$

and $\frac{1+\rho}{2} \geq \sqrt{\rho}$ for all $\rho \in [0,1]$.

We now combine Lemma 3.31 with Theorem 3.28 to get

$$\texttt{B}^\star(\mathcal{C},\ell) = \max\left\{1 - \frac{\mu}{2}, \texttt{B}(\mathcal{C},\ell)\right\} = \max\left\{\frac{1+\rho}{2} - \frac{1}{2n}, \sqrt{\rho}\right\} \leq \frac{1+\rho}{2}.$$

Thus, taking $\texttt{B}^\star(\mathcal{C},\ell) := \frac{1+\rho}{2}$, we get $\delta < 1 - \texttt{B}^\star = \frac{1-\rho}{2}$, so by Theorem 3.28

$$\texttt{err}^\star = \texttt{err} = \frac{2^m(\ell-1)}{|\mathbb{F}|\rho}.$$

□

Mutual correlated agreement is necessary for proving the round-by-round soundness

of WHIR. We define WHIR for arbitrary $\delta \in [0,1]$ in Chapter 5, however, security for δ between the unique decoding distance and the Johnson bound will rely on the following conjecture.

Conjecture 3.33 ([ACFY25] Conjecture 4.12)**.** *The function* $\texttt{Gen}(\ell;\alpha) = (1,\alpha,\ldots,\alpha^{\ell-1})$ *is a proximity generator with mutual correlated agreement with proximity bound* $\texttt{B}^\star$ *and error* $\texttt{err}^\star$ *for every smooth Reed–Solomon code* $\mathrm{RS}[\mathbb{F},\mathcal{L},m]$ *($\rho = 2^m/|\mathcal{L}|$). For $\delta < 1-\sqrt{\rho}$, we have*

$$\texttt{B}^\star(\mathcal{C},\ell) := \sqrt{\rho},$$

and

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

The next lemma shows that proximity generators with mutually correlated agreement "commute" with list decoding: applying the proximity generator and then list decoding gives the same result as first list decoding and then applying the proximity generator on all possible lists of results. This will of course only be relevant if we assume Conjecture 3.33.

It follows from the definition of the interleaved code $\mathcal{C}^\ell$ (Definition 3.14) that $\boldsymbol{u} \in \mathrm{List}(\mathcal{C}^\ell,(f_1,\ldots,f_\ell),\delta)$ if and only if there are at least $(1-\delta)n$ positions where $\boldsymbol{u}$ agrees with each of f_i.

If $\boldsymbol{u} \in \mathcal{C}^\ell$ and

$$\boldsymbol{u} = \big((u_{1,1},\ldots,u_{\ell,1}),\ldots,(u_{1,n},\ldots,u_{\ell,n})\big),$$

then for each $i \in [\ell]$, we define

$$u_i := (u_{i,1},\ldots,u_{i,n}) \in \mathcal{C}.$$

Importantly, u_i *is not* the i-th coordinate of $\boldsymbol{u}$.

In the following lemma, α is a seed for the proximity generator $\texttt{Gen}$.

Lemma 3.34 ([ACFY25] Lemma 4.13)**.** *Let $\mathcal{C} \le \mathbb{F}^n$ be a linear code with relative minimum weight μ and let* $\texttt{Gen}$ *be a proximity generator for $\mathcal{C}$ with mutual correlated agreement with proximity bound* $\texttt{B}^\star$ *and error* $\texttt{err}^\star$*. Let $w_\star \in \mathbb{N}$. Then, for every*

$f_1, \ldots, f_\ell \in \mathbb{F}^n$ *and* $\delta \in (0, \min\{\mu, 1 - \mathtt{B}^\star(\mathcal{C}, \ell)\})$*:*

$$\Pr_{\substack{\alpha \leftarrow \$\mathbb{B}^{w_\star} \\ \boldsymbol{r} = \mathtt{Gen}(\ell;\alpha)}} \left[\mathrm{List}\left(\mathcal{C}, \sum_{j\in[\ell]} r_j f_j, \delta\right) \neq \left\{ \sum_{j\in[\ell]} r_j u_j : \boldsymbol{u} \in \mathrm{List}(\mathcal{C}^\ell, (f_1, \ldots, f_\ell), \delta) \right\} \right]$$
$$\leq \mathtt{err}^\star(\mathcal{C}, \ell, \delta).$$

Proof. Let $\boldsymbol{r} \in \mathbb{F}^\ell$ and let

$$T_{\boldsymbol{r}} := \mathrm{List}\left(\mathcal{C}, \sum_{j\in[\ell]} r_j f_j, \delta\right),$$
$$S_{\boldsymbol{r}} := \left\{ \sum_{j\in[\ell]} r_j u_j : \boldsymbol{u} \in \mathrm{List}(\mathcal{C}^\ell, (f_1, \ldots, f_\ell), \delta) \right\}.$$

We want to show that with high probability, $T_{\boldsymbol{r}} = S_{\boldsymbol{r}}$, to which end we prove two inclusions.

- $S_{\boldsymbol{r}} \subseteq T_{\boldsymbol{r}}$. Let $u \in S_{\boldsymbol{r}}$, then $u = \sum_{j\in\ell} r_j u_j$ for $\boldsymbol{u} \in \mathrm{List}(\mathcal{C}^\ell, (f_1, \ldots, f_\ell), \delta)$. By the definition of the interleaved code, this means that there exists a set $S \subseteq [n]$ such that $|S| \geq (1-\delta)n$ and for every $j \in [\ell]$ and every $x \in S$, $f_j(x) = u_j(x)$. But then for every $x \in S$ it must hold that

  $$\sum_{j\in[\ell]} r_j f_j(x) = \sum_{j\in[\ell]} r_j u_j(x) = u(x)$$

  and thus $\Delta(u, \sum_{j\in[\ell]} r_j f_j) \leq \delta$ and $u \in T_{\boldsymbol{r}}$.

- $T_{\boldsymbol{r}} \subseteq S_{\boldsymbol{r}}$ with high probability. Fix $\boldsymbol{r} \leftarrow \mathtt{Gen}(\ell;\alpha)$ such that for every $W \subseteq [n]$ with $|W| \geq (1-\delta)n$ either of the following holds:

  1. no code word of $\mathcal{C}$ agrees with $\sum_{j\in[\ell]} r_j f_j$ on all of W:

     $$\forall u \in \mathcal{C}, \exists x \in W, u(x) \neq \sum_{j\in[\ell]} r_j f_j(x),$$

2. each f_i agrees with some code word of $\mathcal{C}$ on all of W:

$$\forall i \in [\ell], \exists u_i \in \mathcal{C}, \forall x \in W, u_i(x) = f_i(x).$$

Since $\texttt{Gen}$ is a proximity generator for $\mathcal{C}$ with mutual correlated agreement with error $\texttt{err}^\star$, there are at least $(1 - \texttt{err}^\star(\mathcal{C}, \ell, \delta))2^{w_\star}$ choices of α that result in such an $\boldsymbol{r}$.

Consider $u \in T_{\boldsymbol{r}}$. By definition of $T_{\boldsymbol{r}}$, $u \in \mathcal{C}$ and $\Delta\left(u, \sum_{j\in[\ell]} r_j f_j\right) \leq \delta$. This means that Item 1 does not hold, so it must be that Item 2 holds. Therefore there exist $u_1, \ldots, u_\ell \in \mathcal{C}$ with $u_i(x) = f_i(x)$ for every $x \in W$. Let $S \subseteq [n]$ be the maximal set for which $u_i(x) = f_i(x)$ for every $x \in S$, and observe that by definition $W \subseteq S$, meaning that $|S| \geq |W| \geq (1 - \delta)n$.

By definition, for every $x \in S$, we have

$$u(x) = \sum_{j\in[\ell]} r_j f_j(x) = \sum_{j\in[\ell]} r_j u_j(x)$$

and thus $\Delta\left(u, \sum_{j\in[\ell]} r_j u_j\right) \leq \delta < \mu$. Since u and $\sum_{j\in[\ell]} r_j u_j$ are both code words, this implies that $u = \sum_{j\in[\ell]} r_j u_j$, and as a result $u \in S_{\boldsymbol{r}}$. Observe that the above analysis is true for every $u \in T_{\boldsymbol{r}}$ simultaneously, as long as $\boldsymbol{r}$ has the mutual correlated agreement property. As mentioned before, there are at least $(1 - \texttt{err}^\star(\mathcal{C}, \ell, \delta))2^{w_\star}$ such choices of α out of $2^{w_\star}$ possible choices of α. This concludes the proof.

□

3.5 Block relative distance

We define a new type of distance for constrained Reed–Solomon codes. It reflects how many blocks of coordinates differ between words, instead of individual coordinates.

Definition 3.35 ([ACFY25] Definition 4.16)**.** *Let $\mathcal{L} \subseteq \mathbb{F}$ be a smooth evaluation*

domain and $k \in \mathbb{N}$ be a folding parameter. For $z \in \mathcal{L}^{(2^k)}$, define

$$\mathrm{Block}(\mathcal{L}, k, z) := \{y \in \mathcal{L} : y^{2^k} = z\}.$$

So $\mathrm{Block}(\mathcal{L}, k, z)$ consists of all the 2^k-th roots of z in $\mathcal{L}$, of which there are exactly 2^k because $\mathcal{L}$ is smooth. To calculate $\mathrm{Fold}(f, \boldsymbol{\alpha})(z)$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$, we need to know the values of f at $\mathrm{Block}(\mathcal{L}, k, z)$.

Definition 3.36 ([ACFY25] Definition 4.17)**.** *Let $\mathcal{C} := \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{\omega}, \sigma]$ be a constrained Reed–Solomon code and let $f, g \colon \mathcal{L} \to \mathbb{F}$. We define the **$k$-wise block relative distance** as*

$$\Delta_b(\mathcal{C}, k, f, g) = \frac{\left|\left\{z \in \mathcal{L}^{(2^k)} : \exists y \in \mathrm{Block}(\mathcal{L}, k, z), f(y) \neq g(y)\right\}\right|}{\left|\mathcal{L}^{(2^k)}\right|}.$$

For $S \subseteq \mathbb{F}^{\mathcal{L}}$, we let $\Delta_b(\mathcal{C}, k, f, S) := \min_{g \in S} \Delta_b(\mathcal{C}, k, f, g)$.

By definition, $\Delta_b(\mathcal{C}, 0, f, g)$ coincides with the usual relative Hamming distance $\Delta(f, g)$ for any code $\mathcal{C}$. We define block list decoding of a code word.

Definition 3.37 ([ACFY25] Definition 4.18)**.** *For $\mathcal{C} = \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{\omega}, \sigma]$, proximity parameter $\delta \in [0, 1]$, and $f \colon \mathcal{L} \to \mathbb{F}$, we let*

$$\mathrm{List}_b[\mathcal{C}, k, f, \delta] := \{u \in \mathcal{C} : \Delta_b(\mathcal{C}, k, f, u) \leq \delta\}$$

denote the list of code words in $\mathcal{C}$ within relative block distance at most δ from f.

Example 3.38. *Let $\mathbb{F}_{17} \supseteq \mathcal{L} = \{3, 5, 6, 7, -7, -6, -5, -3\}$ and $\mathcal{C} = \mathrm{RS}[\mathbb{F}_{17}, \mathcal{L}, 2]$. Let $\hat{f} = X^3 + X + 1$ and $\hat{g} = 3X^2 + 3X - 5$ be code words of $\mathcal{C}$. We fix the ordering $(3, 5, 6, 7, -7, -6, -5, -3)$ of the elements of $\mathcal{L}$. This lets us write f and g out coordinate-wise:*

x	3	5	6	7	-7	-6	-5	-3
$\hat{f}(x)$	-3	-5	2	-6	8	0	7	5
$\hat{g}(x)$	-3	0	2	-7	2	0	4	-4

We calculate the distances between these words.

- $\Delta_b(\mathcal{C}, 0, \hat{f}, \hat{g}) = \Delta(\hat{f}, \hat{g}) = \dfrac{5}{8}$.
- *We have $\mathcal{L}^{(2)} = \{2, 8, -8, -2\}$. We calculate the corresponding square roots:*

$(\pm 6)^2 = 2$, $(\pm 5)^2 = 8$, $(\pm 3)^2 = -8$ *and* $(\pm 7)^2 = -2$. *We can now calculate the block distance:*

$$\Delta_b(\mathcal{C}, 1, \hat{f}, \hat{g}) = \frac{|\{8, -8, -2\}|}{|\{2, 8, -8, -2\}|} = \frac{3}{4}.$$

- *Analogously, we get*

$$\Delta_b(\mathcal{C}, 2, \hat{f}, \hat{g}) = \frac{|\{4, -4\}|}{|\{4, -4\}|} = 1$$

and

$$\Delta_b(\mathcal{C}, 3, \hat{f}, \hat{g}) = \frac{|\{-1\}|}{|\{-1\}|} = 1.$$

We see from the example above that by increasing the value of k in $\Delta_b(\mathcal{C}, k, \hat{f}, \hat{g})$, the relative block distance increases. This is generally true. The following is a generalization of Claim 4.19. from [ACFY25].

Proposition 3.39. *Let* $\mathcal{C} = \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{\omega}, \sigma]$, $k, l \in \mathbb{N}$ *with* $k \leq l$, *and* $f, g \colon \mathcal{L} \to \mathbb{F}$. *Then* $\Delta_b(\mathcal{C}, k, f, g) \leq \Delta_b(\mathcal{C}, l, f, g)$ *and consequently, for any* $\delta \in [0, 1]$, *we have* $\mathrm{List}_b[\mathcal{C}, l, f, \delta] \subseteq \mathrm{List}_b[\mathcal{C}, k, f, \delta]$.

Proof. As $\mathcal{L}$ is smooth, for every element $z \in \mathcal{L}^{(2^l)}$, the corresponding block $\mathrm{Block}(\mathcal{L}, l, z)$ has 2^l elements. For any $z' \in \mathcal{L}^{(2^k)}$, $|\mathrm{Block}(\mathcal{L}, k, z')| = 2^k$.

Let S and T denote the sets of blocks of size 2^l and 2^k, respectively, where f and g agree: $S := \{z \in \mathcal{L}^{(2^l)} : \forall y \in \mathrm{Block}(\mathcal{L}, l, z), f(y) = g(y)\}$ and $T := \{z \in \mathcal{L}^{(2^k)} : \forall y \in \mathrm{Block}(\mathcal{L}, k, z), f(y) = g(y)\}$. By the definition of block distance, $\Delta_b(\mathcal{C}, l, f, g) = 1 - \dfrac{|S|}{\left|\mathcal{L}^{(2^l)}\right|}$ and $\Delta_b(\mathcal{C}, k, f, g) = 1 - \dfrac{|T|}{\left|\mathcal{L}^{(2^k)}\right|}$.

Let $z \in \mathcal{L}^{(2^k)}$. Then $z^{2^{l-k}} \in \mathcal{L}^{(2^l)}$ and if $z^{2^{l-k}} \in S$, then $\forall y \in \mathrm{Block}(\mathcal{L}, k, z)$, $f(y) = g(y)$. Thus, every element of S corresponds to 2^{l-k} elements of T. There may, however, be more elements in T, so $|T| \geq 2^{l-k}|S|$. This yields the desired inequality:

$$\begin{aligned}\Delta_b(\mathcal{C}, k, f, g) &= 1 - \frac{|T|}{|\mathcal{L}^{(2^k)}|} \leq 1 - \frac{2^{l-k}|S|}{|\mathcal{L}^{(2^k)}|} = 1 - \frac{2^{l-k}|S|}{2^{l-k}|\mathcal{L}^{(2^k)}|} \\ &= 1 - \frac{|S|}{|\mathcal{L}^{(2^l)}|} = \Delta_b(\mathcal{C}, l, f, g).\end{aligned}$$

The implication that $\mathrm{List}_b[\mathcal{C}, l, f, \delta] \subseteq \mathrm{List}_b[\mathcal{C}, k, f, \delta]$ for any $\delta \in [0, 1]$ is an immediate consequence. □

Since $\Delta(f, g) = \Delta_b(\mathcal{C}, 0, f, g)$, the above proves that the relative Hamming distance is a lower bound for all block relative distances.

3.6 Folding preserves list decoding

In this chapter we prove that if we have mutual correlated agreement, then, with high probability, the list-decoding of a folding of a function f is equal to the folding of the list-decoding of f. This is an integral result for the soundness analysis of WHIR.

We start by proving the following lemma, based on [ACFY25] (Lemma 4.21, Claims 4.22, 4.23). We have corrected some inaccuracies in the original proof as well as improved the exposition while preserving the original statement and overall logic of the result.

Lemma 3.40 ([ACFY25] Lemma 4.21, Claims 4.22, 4.23)**.** [2] *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ be a smooth Reed–Solomon code, and $k \in [m]$ a parameter. Let $\mathcal{C}' := \mathrm{RS}[\mathbb{F}, \mathcal{L}^{(2)}, m-1]$ and let $\mathtt{Gen}(2; \alpha) := (1, \alpha)$ be a proximity generator with mutual correlated agreement for the code $\mathcal{C}'$ with proximity bound $\mathtt{B}^\star$ and error $\mathtt{err}^\star$. Let $\mathtt{B}^\star(\mathcal{C}', 2) \geq 2^m/|\mathcal{L}| =: \rho$. Then for every $f \colon \mathcal{L} \to \mathbb{F}$ and $\delta \in (0, 1 - \mathtt{B}^\star(\mathcal{C}', 2))$,*

$$\Pr_{\alpha \leftarrow \$\mathbb{F}} \left[\begin{array}{c} \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \alpha) \\ \neq \\ \mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \alpha), \delta) \end{array} \right] < \mathtt{err}^\star(\mathcal{C}', 2, \delta).$$

Proof. We prove that with high probability, there are two inclusions between the sets $\mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \alpha)$ and $\mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \boldsymbol{\alpha}), \delta)$.

- We first show that $\mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \alpha) \subseteq \mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \boldsymbol{\alpha}), \delta)$ for any $\alpha \in \mathbb{F}$. Fix $\alpha \in \mathbb{F}$ and let $u \in \mathrm{List}_b(\mathcal{C}, k, f, \delta)$.

$$Z := \left\{ z \in \mathcal{L}^{(2^k)} : \forall x \in \mathrm{Block}(\mathcal{L}, k, z), u(x) = f(x) \right\}.$$

[2] We removed the requirement that $\mathcal{C}$ have mutual correlated agreement, explicitly included the condition $\mathtt{B}^\star(\mathcal{C}', 2) \geq \rho$, and corrected several typos in the original proof.

Let $z \in Z$ and $y \in \text{Block}(\mathcal{L}^{(2)}, k-1, z)$. Then $y = x^2$ for some $x \in \text{Block}(\mathcal{L}, k, z)$. If $x \in \text{Block}(\mathcal{L}, k, z)$, then $x^{2^k} = z = (-x)^{2^k}$ and thus also $-x \in \text{Block}(\mathcal{L}, k, z)$. Now, by definition of Z, we have $f(x) = u(x)$ and $f(-x) = u(-x)$. Hence,

$$\begin{aligned}\text{Fold}(f, \alpha)(y) &= \frac{f(x) + f(-x)}{2} + \alpha \cdot \frac{f(x) - f(-x)}{2x} \\ &= \frac{u(x) + u(-x)}{2} + \alpha \cdot \frac{u(x) - u(-x)}{2x} = \text{Fold}(u, \alpha)(y).\end{aligned}$$

Thus, for all $z \in Z$, $\text{Fold}(f, \alpha)$ and $\text{Fold}(u, \alpha)$ agree on all elements of $\text{Block}(\mathcal{L}^{(2)}, k-1, z)$. In other words, if u and f agree on $\text{Block}(\mathcal{L}, k, z)$, then $\text{Fold}(u, \alpha)$ and $\text{Fold}(f, \alpha)$ agree on $\text{Block}(\mathcal{L}^{(2)}, k-1, z)$. It follows that

$$\Delta_b(\mathcal{L}^{(2)}, k-1, \text{Fold}(f, \alpha), \text{Fold}(u, \alpha)) \leq \Delta_b(\mathcal{L}, k, f, u) \leq \delta.$$

The proof of this inclusion concludes, since by Proposition 3.25, $\text{Fold}(u, \alpha) \in \text{RS}[\mathbb{F}, \mathcal{L}^{(2)}, m-1] = \mathcal{C}'$, and thus $\text{Fold}(u, \alpha) \in \text{List}_b(\mathcal{C}', k-1, \text{Fold}(f, \alpha), \delta)$.

- We now prove that $\Pr_{\alpha \leftarrow \$ \mathbb{F}} \left[\begin{array}{c} \text{List}_b(\mathcal{C}', k-1, \text{Fold}(f, \alpha), \delta) \\ \not\subseteq \\ \text{Fold}(\text{List}_b(\mathcal{C}, k, f, \delta), \alpha) \end{array} \right] < \texttt{err}^\star(\mathcal{C}', 2, \delta)$.

  Let $f_0, f_1 : \mathcal{L}^{(2)} \to \mathbb{F}$ be defined as $f_0(x^2) := \dfrac{f(x) + f(-x)}{2}$ and $f_1(x^2) := \dfrac{f(x) - f(-x)}{2x}$ and fix $\alpha \in \mathbb{F}$ for which there is no set $W \subseteq \mathcal{L}^{(2)}$ with $|W| \geq (1-\delta)|\mathcal{L}^{(2)}|$ for which the following conditions both hold:

  1. there exists $u \in \mathcal{C}'$ such that $u(y) = f_0(y) + \alpha f_1(y)$ for every $y \in W$, and
  2. there exists $b \in \{0, 1\}$ where for every $u \in \mathcal{C}'$ there exists $y \in W$ such that $u(y) \neq f_b(y)$.

  Since $\text{Gen}(2; \alpha) = (1, \alpha)$ is a proximity generator for $\mathcal{C}'$ with mutual correlated agreement, error $\texttt{err}^\star$ and bound $\texttt{B}^\star$, there are at least $(1 - \texttt{err}^\star)|\mathbb{F}|$ such choices of α.

  Suppose that $v \in \text{List}_b(\mathcal{C}', k-1, \text{Fold}(f, \alpha), \delta)$. Then note that $\left(\mathcal{L}^{(2)}\right)^{(2^{k-1})} =$

$\mathcal{L}^{(2^k)}$ and let

$$\begin{aligned} Z :=& \{z \in \mathcal{L}^{(2^k)} : \forall y \in \mathrm{Block}(\mathcal{L}^{(2)}, k-1, z), f_0(y) + \alpha f_1(y) = v(y)\} \\ =& \{z \in (\mathcal{L}^{(2)})^{(2^{k-1})} : \forall y \in \mathrm{Block}(\mathcal{L}^{(2)}, k-1, z), \mathrm{Fold}(f, \alpha)(y) = v(y)\}. \end{aligned}$$

Since $v \in \mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \alpha), \delta)$, we deduce that

$$|Z| \geq (1-\delta)|(\mathcal{L}^{(2)})^{(2^{k-1})}| = (1-\delta)|\mathcal{L}^{(2^k)}|.$$

Let $W := \bigcup_{z \in Z} \mathrm{Block}(\mathcal{L}^{(2)}, k-1, z)$. Since each block has 2^{k-1} elements and the blocks do not intersect, we have $|W| = 2^{k-1}|Z|$ and since $|Z| \geq (1-\delta)|\mathcal{L}^{(2^k)}|$, we conclude $|W| \geq (1-\delta)|\mathcal{L}^{(2)}|$.

Since W fills the condition in Item 1, Item 2 must not hold for our choice of α. Hence, there exist $u_0, u_1 \in \mathcal{C}'$ such that $f_0(y) = u_0(y)$ and $f_1(y) = u_1(y)$ for each $y \in W$. Since $\mathcal{L}^{(2)}$ is smooth, we can write $u_0(y) = \hat{u}_0(\mathrm{pow}(y, m-1))$ and $u_1(y) = \hat{u}_1(\mathrm{pow}(y, m-1))$.

Define the multilinear polynomial

$$\hat{q}(X_1, \ldots, X_m) = \hat{u}_0(X_2, \ldots, X_m) + X_1 \hat{u}_1(X_2, \ldots, X_m).$$

Let $y \in W$ and $x \in \mathcal{L}$ with $y = x^2$. Then

$$\begin{aligned} q(x) :=& \hat{q}(\mathrm{pow}(x, m)) \\ =& \hat{q}(x, \mathrm{pow}(y, m-1)) \\ =& \hat{u}_0(\mathrm{pow}(y, m-1)) + x\hat{u}_1(\mathrm{pow}(y, m-1)) \\ =& f_0(y) + x f_1(y) \\ =& \frac{f(x) + f(-x)}{2} + x \cdot \frac{f(x) - f(-x)}{2x} \\ =& f(x). \end{aligned}$$

By the definition of W, f and q agree on every x such that $x^{2^k} \in Z$. Since $|Z| \geq (1-\delta)|\mathcal{L}^{(2^k)}|$, we have $q \in \mathrm{List}_b(\mathcal{C}, k, f, \delta)$.

The above equality also shows that for every $y \in W$, $v(y) = \mathrm{Fold}(q, \alpha)(y)$. Since δ is smaller than the relative minimum distance of $\mathcal{C}'$ ($\delta < 1 - \rho$) and

v and $\mathrm{Fold}(q, \alpha)$ are both code words, we have $v = \mathrm{Fold}(q, \alpha)$.

This implies that $v \in \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \alpha)$. Since the above arguments hold for every $v \in \mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \alpha), \delta)$, we infer that for this choice of α,

$$\mathrm{List}_b(\mathcal{C}', k-1, \mathrm{Fold}(f, \alpha), \delta) \subseteq \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \alpha).$$

We finish by recalling that there are at least $(1 - \mathtt{err}^\star)|\mathbb{F}|$ such choices for α.

□

Theorem 3.41 ([ACFY25] Theorem 4.20). [3] *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ be a smooth Reed–Solomon code and $k \in [m]$. For $0 \le i \le k$ let $\mathcal{C}^{(i)} := \mathrm{RS}[\mathbb{F}, \mathcal{L}^{(2^i)}, m-i]$. Let $\mathtt{Gen}(\ell; \alpha) = \left(1, \alpha, \ldots, \alpha^{\ell-1}\right)$ be a proximity generator with mutual correlated agreement for the codes $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}$ with proximity bound $\mathtt{B}^\star$ and error $\mathtt{err}^\star$ with $\min_{i \in [k]}\{\mathtt{B}^\star(\mathcal{C}^{(i)}, 2)\} \ge 2^m/|\mathcal{L}| =: \rho$.*

Then for every $f \colon \mathcal{L} \to \mathbb{F}$ and $\delta \in \left(0, 1 - \max_{i \in [k]} \left\{\mathtt{B}^\star\left(\mathcal{C}^{(i)}, 2\right)\right\}\right)$,

$$\Pr_{\boldsymbol{\alpha} \leftarrow \$\mathbb{F}^k}\left[\begin{array}{c} \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \boldsymbol{\alpha}) \\ \neq \\ \mathrm{List}(\mathcal{C}^{(k)}, \mathrm{Fold}(f, \boldsymbol{\alpha}), \delta) \end{array}\right] < \mathtt{err}^{(k)}(\mathcal{C}, \delta)$$

where $\mathtt{err}^{(k)}(\mathcal{C}, \delta) := \sum_{i=1}^{k} \mathtt{err}^*(\mathcal{C}^{(i)}, 2, \delta)$.

Proof. We say that a vector $(\alpha_1, \ldots, \alpha_i) \in \mathbb{F}^i$ is good if

$$\mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), (\alpha_1, \ldots, \alpha_i)) = \mathrm{List}_b(\mathcal{C}^{(i)}, k-i, \mathrm{Fold}(f, (\alpha_1, \ldots, \alpha_i)), \delta).$$

The length 0 vector is good in the sense that by doing no folding the two sets above are identical. Suppose that $(\alpha_1, \ldots, \alpha_{i-1}) \in \mathbb{F}^{i-1}$ is good for some $0 < i \le k$, and

[3] We added the requirement $\min_{i \in [k]}\{\mathtt{B}^\star(\mathcal{C}^{(i)}, 2)\} \ge \rho$ and corrected several index errors in the original proof.

let $g := \mathrm{Fold}(f, (\alpha_1, \ldots, \alpha_{i-1}))$. By Lemma 3.40, we have

$$\Pr_{\alpha_i \leftarrow \$ \mathbb{F}} \left[\begin{array}{c} \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}^{(i-1)}, k-i+1, g, \delta), \alpha_i) \\ \neq \\ \mathrm{List}_b(\mathcal{C}^{(i)}, k-i, \mathrm{Fold}(g, \alpha_i), \delta) \end{array} \right] < \mathtt{err}^\star(\mathcal{C}^{(i)}, 2, \delta). \tag{6}$$

Fix α_i such that the sets in the above event are equal. Then $(\alpha_1, \ldots, \alpha_i)$ is good, since

$$\begin{aligned}
&\mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), (\alpha_1, \ldots, \alpha_i)) \\
&= \mathrm{Fold}(\mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), (\alpha_1, \ldots, \alpha_{i-1})), \alpha_i) && \text{definition of Fold,} \\
&= \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}^{(i-1)}, k-i+1, \mathrm{Fold}(f, (\alpha_1, \ldots, \alpha_{i-1})), \delta), \alpha_i) \\
& && (\alpha_1, \ldots, \alpha_{i-1}) \text{ is good,} \\
&= \mathrm{List}_b(\mathcal{C}^{(i)}, k-i, \mathrm{Fold}(\mathrm{Fold}(f, (\alpha_1, \ldots, \alpha_{i-1})), \alpha_i), \delta) && \text{the sets in (6) equal,} \\
&= \mathrm{List}_b(\mathcal{C}^{(i)}, k-i, \mathrm{Fold}(f, (\alpha_1, \ldots, \alpha_i)), \delta) && \text{definition of Fold.}
\end{aligned}$$

Thus,

$$\Pr_{\alpha_i \leftarrow \$ \mathbb{F}}[(\alpha_1, \ldots, \alpha_i) \text{ is not good} \mid (\alpha_1, \ldots, \alpha_{i-1}) \text{ is good}] < \mathtt{err}^\star(\mathcal{C}^{(i)}, 2, \delta).$$

We conclude that

$$\begin{aligned}
&\Pr_{\boldsymbol{\alpha} \leftarrow \$ \mathbb{F}^k} \left[\begin{array}{c} \mathrm{Fold}(\mathrm{List}_b(\mathcal{C}, k, f, \delta), \boldsymbol{\alpha}) \\ \neq \\ \mathrm{List}(\mathcal{C}^{(k)}, \mathrm{Fold}(f, \boldsymbol{\alpha}), \delta) \end{array} \right] \\
&= \Pr_{\alpha_1, \ldots, \alpha_k \leftarrow \$ \mathbb{F}^k}[(\alpha_1, \ldots, \alpha_k) \text{ is not good}] \\
&\leq \sum_{i=1}^{k} \Pr_{\alpha_i \leftarrow \$ \mathbb{F}}[(\alpha_1, \ldots, \alpha_i) \text{ is not good} \mid (\alpha_1, \ldots, \alpha_{i-1}) \text{ is good}] \\
&< \sum_{i=1}^{k} \mathtt{err}^\star(\mathcal{C}^{(i)}, 2, \delta).
\end{aligned}$$

□

3.7 Out-of-domain sampling

Following [ACFY25], we generalize the results from [ACFY24] concerning the probability that two distinct code words from a list-decoding set agree at a randomly chosen point. This will later be used to bound the error in the out-of-domain sampling stage of the WHIR protocol.

Lemma 3.42 ([ACFY25] Lemma 4.24)**.** *Let $f\colon \mathcal{L} \to \mathbb{F}$ be a function, $m \in \mathbb{N}$ a number of variables, $s \in \mathbb{N}$ a repetition parameter, and $\delta \in [0,1]$ a distance parameter. The following statements are equivalent for every $\boldsymbol{r}_1, \dots, \boldsymbol{r}_s \in \mathbb{F}^m$.*

- *There exist distinct $u, u' \in \mathrm{List}(\mathrm{RS}[\mathbb{F}, \mathcal{L}, m], f, \delta)$ such that, for every $i \in [s]$, $\hat{u}(\boldsymbol{r}_i) = \hat{u}'(\boldsymbol{r}_i)$.*
- *There exist $\sigma_1, \dots, \sigma_s \in \mathbb{F}$ such that*

$$|\,\mathrm{List}(\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (Z\,\mathtt{eq}(\boldsymbol{r}_1, \cdot), \sigma_1), \dots, (Z\,\mathtt{eq}(\boldsymbol{r}_s, \cdot), \sigma_s)], f, \delta)| \geq 2.$$

Proof. We prove each direction.

- For $i \in [s]$, let $\sigma_i := \hat{u}(\boldsymbol{r}_i)$. Then, by Definition 3.21, Definition 2.5 and Definition 3.22

$$u, u' \in \mathrm{List}(\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (Z\,\mathtt{eq}(\boldsymbol{r}_1, \cdot), \sigma_1), \dots, (Z\,\mathtt{eq}(\boldsymbol{r}_s, \cdot), \sigma_s)], f, \delta),$$

  since $u, u' \in \mathrm{List}(\mathrm{RS}[\mathbb{F}, \mathcal{L}, m])$. As u, u' are distinct, this direction follows.

- Let $u, u' \in \mathrm{List}(\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (Z\,\mathtt{eq}(\boldsymbol{r}_1, \cdot), \sigma_1), \dots, (Z\,\mathtt{eq}(\boldsymbol{r}_s, \cdot), \sigma_s)], f, \delta)$ be two distinct code words which exist by hypothesis. But this means that $u, u' \in \mathrm{List}(\mathrm{RS}[\mathbb{F}, \mathcal{L}, m], f, \delta)$, and, by Definition 2.5, it must be that, for every $i \in [s], \hat{u}(\boldsymbol{r}_i) = \hat{u}'(\boldsymbol{r}_i)$.

□

Using Lemma 3.42, we restate Lemma 4.5 from [ACFY24] as follows.

Lemma 3.43 ([ACFY25] Lemma 4.5)**.** *Let $f : \mathcal{L} \to \mathbb{F}$ be a function, $m \in \mathbb{N}$ be a number of variables, $s \in \mathbb{N}$ be a repetition parameter, and $\delta \in [0,1]$ be a distance*

parameter. If $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ *is* (δ, ℓ)*-list decodable then*

$$\begin{aligned}
&\Pr\left[\begin{array}{c}\exists\, \sigma_1, \ldots, \sigma_s \in \mathbb{F} \ s.t. \\ |\,\mathrm{List}(\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (Z\ \mathtt{eq}(\mathrm{pow}(r_1, m), \cdot), \sigma_1), \ldots, \\ (Z\ \mathtt{eq}(\mathrm{pow}(r_s, m), \cdot), \sigma_s)], f, \delta)| \geq 2\end{array}\right] \\
&= \Pr_{r_1, \ldots, r_s \leftarrow\$ \mathbb{F}}\left[\begin{array}{c}\exists\, \text{distinct } u, u' \in \mathrm{List}(\mathrm{RS}[\mathbb{F}, \mathcal{L}, m], f, \delta) \\ s.t.\ \forall i \in [s], \hat{u}(\mathrm{pow}(r_i, m)) = \hat{u}'(\mathrm{pow}(r_i, m))\end{array}\right] \\
&\leq \frac{\ell^2}{2}\left(\frac{2^m}{|\mathbb{F}|}\right)^s.
\end{aligned}$$

4 IOPs and PIOPs

4.1 IOPs and IOPPs

An **interactive oracle proof** (IOP) is an interactive protocol between a **prover** ($\mathcal{P}$) and a **verifier** ($\mathcal{V}$), introduced in [BCS16]. Since this work has need of a more general concept, called an **interactive oracle proof of proximity** (IOPP), we will start by defining this and then define an IOP as a special case of an IOPP.

Let $\{R(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ be a ternary relation computable in polynomial time. In an IOPP, both $\mathcal{P}$ and $\mathcal{V}$ are given full access to the **instance** $\mathbb{x}$. The prover additionally has access to $\mathbb{y}$, while the verifier only has oracle access to it. The goal of $\mathcal{P}$ is to convince $\mathcal{V}$ that it possesses a **witness** $\mathbb{w}$ such that $R(\mathbb{x}, \mathbb{y}, \mathbb{w}) = 1$.

In a k-round IOPP $(\mathcal{P}, \mathcal{V})$, the interaction consists of k rounds. In the i-th round, the verifier $\mathcal{V}$ sends a message α_i to the prover. The prover responds by sending a string (vector) π_i to an oracle which $\mathcal{V}$ can subsequently query for any locations $q_{i,j} \in |\pi_i|$, $j \in [n_i]$ (n_i denotes the number of queries). The oracle is assumed to be honest, meaning it always behaves as specified and answers consistently with the string provided by the prover. The string $\mathtt{tr}$ consisting of all the messages sent up to a given point in time is called the (partial) **transcript** of the interaction. To have the protocol start with a prover message, the first verifier message α_1 can be considered empty (or just be ignored).

After k rounds, using all the information obtained during the protocol, the verifier outputs a decision bit $b \in \{0, 1\}$, where $b = 1$ denotes acceptance and $b = 0$ denotes rejection. In most IOPP constructions, the verifier's messages are obtained via **public coin**: α_i, $i \in [k]$, are chosen uniformly at random from some set. The prover's responses π_i may depend on all previously exchanged messages and any private data available to him.

An IOP is an IOPP where we take $\mathbb{y} = \perp$. In this case, R can be viewed as a binary relation. See Figure 6 for an illustration of the IOPP protocol.

There are two fundamental parameters of IOPPs: the **proof length** $|\pi| = \sum_{i=1}^{k} |\pi_i|$ which is the sum of the lengths of the prover's messages, and the **query complexity** $q = \sum_{i=0}^{k} n_i$ which is the total number of locations $\mathcal{V}$ queries across all of the

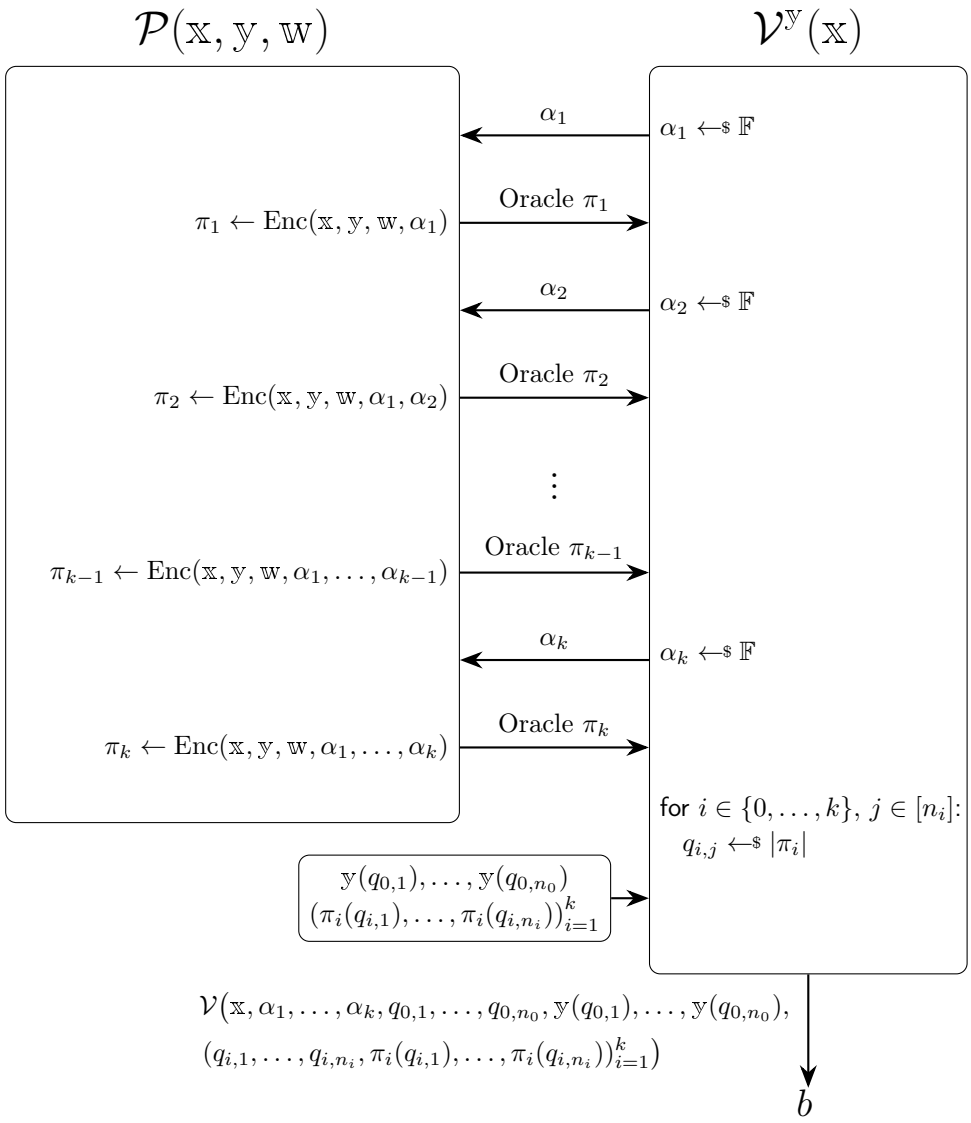

Figure 6: IOPP between prover $\mathcal{P}$ and verifier $\mathcal{V}$.

prover's messages and $\mathtt{y}$ (here, n_0 denotes the number of $\mathcal{V}$'s queries to $\mathtt{y}$ and n_i denotes the number of $\mathcal{V}$'s queries to π_i, $i \in [k]$).

Definition 4.1 (δ-proximity). *Let R be a ternary polynomial-time relation. We say that $(\mathtt{x}, \mathtt{y})$ is δ-**far** from R if $(\mathtt{x}, \mathtt{y}', \mathtt{w}) \notin R$ for all $\mathtt{w}$ and all $\mathtt{y}'$ with $\Delta(\mathtt{y}, \mathtt{y}') \leq \delta$. If $(\mathtt{x}, \mathtt{y})$ is not δ-far from R, we say it is δ-**close** to R.*

We introduce some security notions for IOPPs.

Definition 4.2. *Let $b = \mathcal{V}^{\mathtt{y}, f_1, \ldots, f_k}(\mathtt{x}, \alpha_1, \ldots, \alpha_k)$. We say an IOPP is*

- ***(perfectly) complete** if for all $(\mathtt{x}, \mathtt{y}, \mathtt{w}) \in R$, when honest $\mathcal{V}$ interacts with honest $\mathcal{P}$,*
$$\Pr(b = 1) = 1.$$
- ***sound** if for all unbounded malicious provers $P^\star$ and $(\mathtt{x}, \mathtt{y}) \notin L(R) := \{(\mathtt{x}, \mathtt{y}) : \exists \mathtt{w} \text{ s.t. } (\mathtt{x}, \mathtt{y}, \mathtt{w}) \in R\}$,*
$$\Pr(b = 1) < \mu(\lambda)$$
where λ is the security parameter and μ a negligible function.
- *δ-**sound** if for all unbounded malicious provers $P^\star$ and $(\mathtt{x}, \mathtt{y})$ that are δ-far from R*
$$\Pr(b = 1) < \mu(\lambda).$$

In the case of IOPPs, we have to use δ-soundness instead of regular soundness. The reason is that the amount of queries $\mathcal{V}$ can make to $\mathtt{y}$ is usually considered to be logarithmic in the length of $\mathtt{y}$ but since the prover could cheat in as few as one single coordinate of $\mathtt{y}$, regular soundness would need a linear amount of queries in $\mathtt{y}$. This is why IOPPs are called IOPs of *proximity*: the protocol can by design only check proximity to the relation R, not belonging to the relation itself.

The BCS transformation (to get a zk-SNARK from an IOP) requires another notion of soundness, called *state-restoration soundness*, introduced along with IOPs in [BCS16]. In [CCHLRRW19], a stronger notion of soundness called *round-by-round soundness* was coined. It was proved in the same paper that round-by-round soundness implies having state-restoration soundness. Thus, to prove the security of the hash-based zk-SNARK constructed via proximity testing and the BCS transformation, we prove the round-by-round soundness of the applied IOPP.

Intuitively, round-by-round soundness means that each round of the protocol needs to have a small soundness error on its own, in contrast to only requiring it from the protocol as a whole. In order to define round-by-round soundness, we must first define the notion of a state function. Intuitively, $\texttt{State}$ records whether at a given stage of the interaction, the prover can still steer the interaction toward acceptance: outputting 1 means a prover continuation might lead to acceptance, while 0 means the prover itself cannot, by any message, make acceptance possible.

For a relation $\{R(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$, we say $(\mathbb{x}, \mathbb{y}) \in L(R)$ if and only if $\exists \mathbb{w}, (\mathbb{x}, \mathbb{y}, \mathbb{w}) \in R$.

Definition 4.3 (State function). *Let $(\mathcal{P}, \mathcal{V})$ be an IOPP for a relation $R = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$. A **state function** for $(\mathcal{P}, \mathcal{V})$ is a function $\texttt{State}$ that receives as inputs $\mathbb{x}$, $\mathbb{y}$ and a transcript $\texttt{tr}$ and outputs a bit, and has the following properties.*

- *If $\texttt{tr} = \emptyset$ then $\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr}) = 1$ if and only if $(\mathbb{x}, \mathbb{y}) \in L(R)$.*
- *If $\texttt{tr}$ is a transcript where $\mathcal{P}$ is about to move and $\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr}) = 0$ then, for every prover message π, $\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr} \| \pi) = 0$.*
- *If $\texttt{tr}$ is a full transcript and $\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr}) = 0$ then $\mathcal{V}$ rejects the interaction.*

Definition 4.4 (Round-by-round soundness). *A k-round IOPP $(\mathcal{P}, \mathcal{V})$ for a relation $R = \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ has **round-by-round soundness** with errors $(\varepsilon_1, \ldots, \varepsilon_k)$ if the IOPP has a state function $\texttt{State}$ and an unbounded deterministic extractor $\mathcal{E}$ with the following property: for every transcript $\texttt{tr} = (\alpha_1, \pi_1, \ldots, \alpha_{i-1}, \pi_{i-1})$ and $\mathbb{x}$, $\mathbb{y}$, if*

- $\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr}) = 0$, *and*
- $\Pr_{\alpha_i \leftarrow \mathcal{V}}[\texttt{State}(\mathbb{x}, \mathbb{y}, \texttt{tr} \| \alpha_i) = 1] > \varepsilon_i(\mathbb{x}, \mathbb{y})$,

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

Intuitively, round-by-round soundness means that if $\mathcal{P}$ can, with non-negligible probability, transition from a transcript where the state function outputs 0 to one where it outputs 1, then a valid witness must exist.

An important example of an IOPP is the Reed-Solomon IOPP (RS-IOPP).

Definition 4.5. *Let $\mathcal{C}$ be a smooth* $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ *code,* $u \colon \mathcal{L} \to \mathbb{F}$ *and* $\delta \in [0, 1]$. *A* ***δ-RS-IOPP*** *is a complete and δ-sound IOPP $(\mathcal{P}, \mathcal{V})$ such that* $\mathtt{x} = \mathcal{C}$, $\mathtt{y} = u$ *and* $\mathtt{w} = \perp$. *We say that the IOPP is*

- *complete if in case of honest $\mathcal{P}$:*

$$u \in \mathcal{C} \implies \Pr\Big(\mathcal{V}^{u, f_1, \ldots, f_k}(\mathcal{C}, \alpha_1, \pi_1, \ldots, \alpha_k, \pi_k) = 1\Big) = 1;$$

- *δ-sound if for all unbounded malicious provers $\mathcal{P}^\star$:*

$$\Delta(u, \mathcal{C}) > \delta \implies \Pr\Big(\mathcal{V}^{u, f_1, \ldots, f_k}(\mathcal{C}, \alpha_1, \pi_1, \ldots, \alpha_k, \pi_k) = 1\Big) < \mu(\lambda)$$

  where λ is the security parameter and μ is a negligible function.

4.2 PIOPs

Polynomial interactive oracle proofs (PIOPs) were introduced in 2020 in [BFS20] and [CHMMVW20]. Conceptually, they extend the notion of IOPs by having the prover send bounded degree polynomials (i.e. elements of $\mathbb{F}^{<d}[X]$ for some fixed $d \in \mathbb{N}$) rather than vectors to the oracle. The verifier can subsequently query these polynomials at arbitrary points in the underlying field. This framework provides the verifier with greater flexibility and power, since the field size $|\mathbb{F}_q| = q$ is typically much larger than the length of the message vectors $|\pi_i|$, resulting in a vastly larger space of possible queries.

WHIR uses a multilinear polynomial IOP so we describe this in detail. Let $\{R(\mathtt{x}, \mathtt{w})\}$ be a polynomial-time binary relation. In a k-round multilinear PIOP $(\mathcal{P}, \mathcal{V})$, $\mathcal{P}$ has access to the instance $\mathtt{x}$ and the witness $\mathtt{w}$ whereas the verifier $\mathcal{V}$ only has access to $\mathtt{x}$. In the i-th round, $\mathcal{V}$ sends a challenge $\alpha_i \in \mathbb{F}$ to $\mathcal{P}$. $\mathcal{P}$ responds by sending a multilinear polynomial $\hat{f}_i \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$ to an oracle. $\mathcal{V}$ can henceforth query the oracle for evaluations of $\hat{f}_i$ at vectors of field elements $\boldsymbol{q}_{i,j} \in \mathbb{F}^m$, $j \in [n_i]$, to get $\hat{f}_i(\boldsymbol{q}_{i,j})$. In WHIR, $\mathcal{V}$ samples $q_{i,j} \leftarrow\!\!\$\ \mathbb{F}$ and defines $\boldsymbol{q}_{i,j} := \mathrm{pow}(q_{i,j}, m)$.

The oracle guarantees that the number of variables of $\hat{f}_i$ is not larger than m and that $\hat{f}_i$ is multilinear. After k rounds, $\mathcal{V}$ outputs a decision bit. See Figure 7 for an illustration of the protocol.

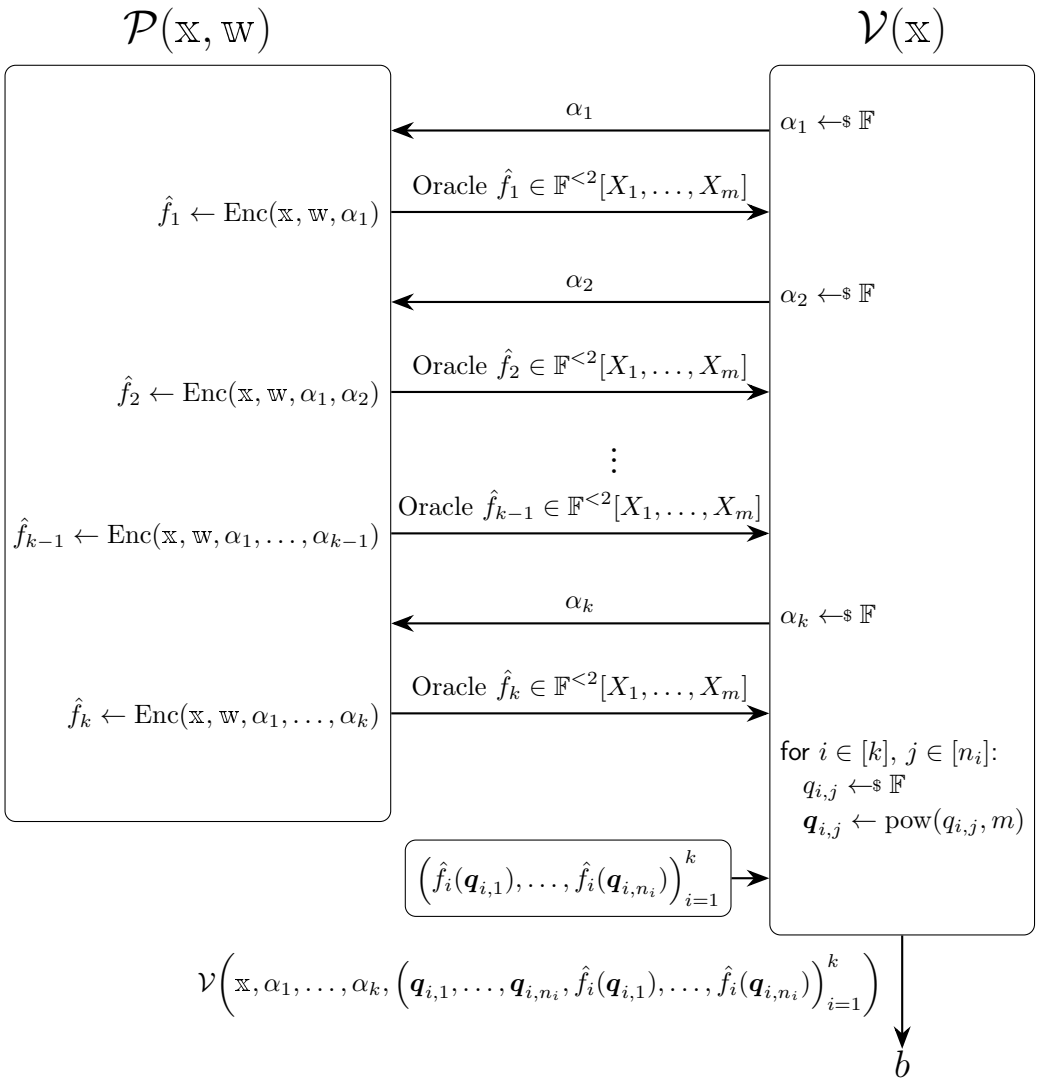

Figure 7: k-round multilinear PIOP between prover $\mathcal{P}$ and verifier $\mathcal{V}$.

4.3 Getting a zk-SNARK from an IOP

In Figure 1 we saw that getting a zk-SNARK using the BCS transformation needs converting the IOP into an interactive proof (IP). From an implementation standpoint, the problem with IOPs is that they need access to a trustworthy third-party oracle. Under cryptographic assumptions, this can be overcome. One common way to achieve this is for $\mathcal{P}$ to use some cryptographic commitment scheme to commit to the vectors π_i and respond to verifier's queries himself. At the end of the protocol, $\mathcal{P}$ sends a proof that the responses to the queries were correct. This converts the IOP into an interactive proof.

In practice, the best commitment scheme to use for this purpose is Merkle trees.

4.3.1 Merkle trees

A Merkle tree is an important cryptographic primitive for committing vectors. The scheme uses a hash function H and its security depends on the collision-resistance of this hash function. The Merkle tree of a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ is a binary tree built in the following steps.

- The leaves of the tree are the individual locations of the vector, i.e. $v_1, \ldots, v_n$.
- In each layer, the nodes are paired up and each pair is concatenated and hashed again to get the parent node. So if x and y are a pair then their parent node is calculated according to the following formula:

  $$\text{Parent}(x, y) = H(x||y).$$

  At some points there may be an odd number of nodes in a layer, then the last three nodes, instead of two, are concatenated and hashed. Alternatively, padding may be used to make the length of $\boldsymbol{v}$ a power of 2.
- This process is continued until the root node, called `com` for commitment, is reached. This node serves as the commitment.

Later, when the verifier asks to open the value v_i (i.e. to check that it was correctly calculated), the prover opens (reveals the element in) the corresponding leaf and exactly the nodes needed for the verifier to calculate `com`.

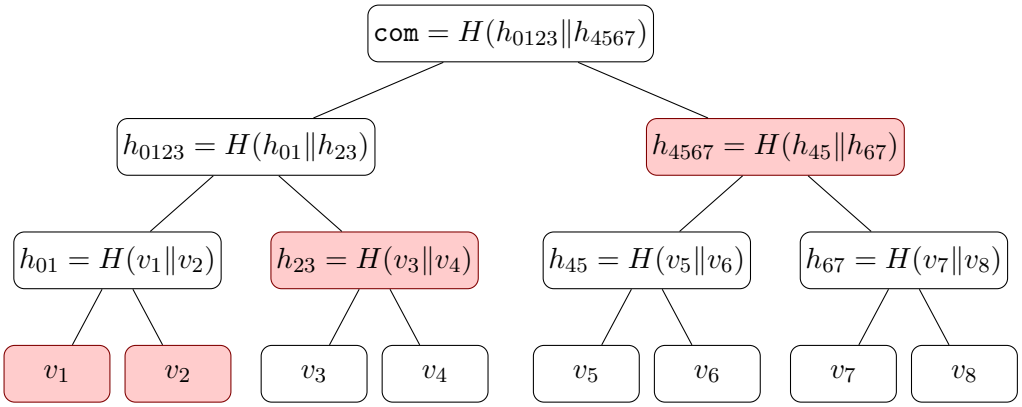

Figure 8: Merkle tree with 8 leaves. In red are nodes that must be revealed to open v_2.

For an example of the nodes which need to be opened to prove that v_2 was calculated correctly, see Figure 8.

4.3.2 Fiat–Shamir

To obtain a non-interactive proof (such as a zk-SNARK) from an interactive protocol, the well-known Fiat–Shamir transformation [FS86] is applied. The idea is to replace the verifier's random challenges with values derived deterministically from a cryptographic hash function. When in the interactive proof the verifier would send a randomly sampled α, here the prover uses a hash of the current transcript of the protocol (which includes all previous messages and the instance) to simulate it. This allows $\mathcal{P}$ to generate the entire proof transcript independently, producing a non-interactive proof that the verifier can later check deterministically by calculating the hashes herself. The security of using Fiat–Shamir in this context was shown in [BCS16]. See Figure 9 for an illustration.

4.4 Compiling a PIOP into an IOP

In the BCS compiler (Figure 1), a PIOP is compiled into an IOP. The idea is to convert the polynomials sent by the PIOP prover into vectors of polynomial evaluations. These vectors are viewed as code words in some code $\mathcal{C}$. The natural choice is to use (smooth) RS codes. However, other codes have been used [BCKL22; BLNR20; HLP24; ZCF24; GLSTW21; XZS22; BCFRRZ25]. As of 2025, one of the

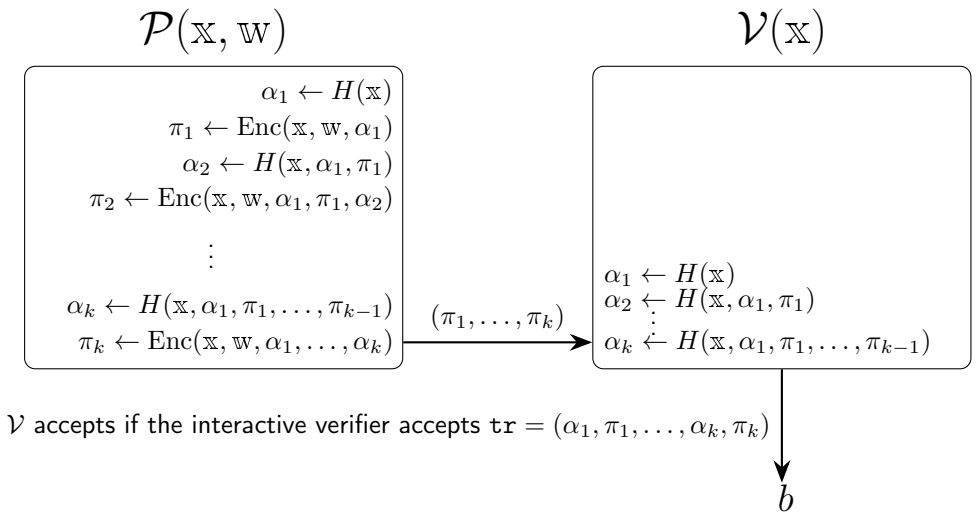

Figure 9: Fiat-Shamir transformation of an interactive protocol

most verifier-efficient IOPP for smooth RS codes is WHIR [ACFY25].

We provide an overview of the compilation here and give more specifics on the use of WHIR in Chapter 5.1.

- We assume that we have a working multilinear PIOP and the goal is to convert this into an IOP without losing soundness. We assume the WHIR paradigm where the verifier samples single elements $q \leftarrow\!\!\$\ \mathbb{F}$ (not vectors) and to pass them as arguments of a multilinear polynomial, we use the vector $\boldsymbol{q} := \mathrm{pow}(q, m)$.

- When the PIOP prover sends an oracle to a multilinear polynomial $\hat{f}_i \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$, the IOP prover instead sends the oracle to the vector f_i: the evaluation of $\hat{f}_i$ on a fixed evaluation domain $\mathcal{L} = (a_1, \ldots, a_n)$. Explicitly,

$$f_i := (\hat{f}_i(\mathrm{pow}(a_1, m)), \ldots, \hat{f}_i(\mathrm{pow}(a_n, m))).$$

  The IOP verifier responds with the same α_{i+1} as the PIOP verifier.

- When the PIOP verifier asks for the evaluation of $\hat{f}_i$ at the point $\boldsymbol{q}_{i,j} = \mathrm{pow}(q_{i,j}, m) \in \mathbb{F}^m$, the PIOP oracle can reply directly. The IOP oracle, however, cannot respond by itself since it does not have access to the polynomial $\hat{f}$, only to its evaluation on $\mathcal{L}$. Instead, the IOP prover replies with

the evaluation and the parties go through an interaction of the IOPP to prove that the evaluation is correct. We show how this is achieved next.

- Let us assume that the PIOP verifier asks for evaluations of $\hat{f}_i$ at the points $\boldsymbol{q}_{i,1}, \ldots, \boldsymbol{q}_{i,n_i}$ and gets the replies $\hat{f}_i(\boldsymbol{q}_{i,j}) = \sigma_{i,j}$, $j = 1, \ldots, n_i$, from the PIOP oracle.

  Since $\hat{f}_i$ is multilinear, in the honest case we have

  $$\sigma_{i,j} = \sum_{\boldsymbol{b} \in \mathbb{B}^m} \hat{f}_i(\boldsymbol{b}) \, \mathsf{eq}(\boldsymbol{q}_{i,j}, \boldsymbol{b})$$

  by Proposition 2.4. Defining $\hat{w}_{i,j}(Z, X_1, \ldots, X_m) := Z \, \mathsf{eq}(\boldsymbol{q}_{i,j}, X_1, \ldots, X_m)$, the IOP verifier can reduce checking the evaluation $\hat{f}_i(\boldsymbol{q}_{i,j}) = \sigma_{i,j}$ to verifying that

  $$f_i \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{w}_{i,j}, \sigma_{i,j}].$$

  Using multi-constrained RS codes, the verifier can reduce checking all evaluations at to checking one code membership

  $$f_i \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_{i,j}, \sigma_{i,j})_{j=1}^{n_i}].$$

- The IOP prover and verifier engage in the WHIR protocol to prove that f_i is δ-close to the code $\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_{i,j}, \sigma_{i,j})_{j=1}^{n_i}]$. If the WHIR interaction is accepted, then with high probability there exists a polynomial $\hat{f}_i'$ that is δ-close to f_i and satisfies all the evaluation constraints.

  If δ is chosen within the unique decoding radius of $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$, then $\hat{f}_i'$ is unique and can be recovered by Reed–Solomon decoding.

  If δ is chosen within the Johnson bound of $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$, then by Theorem 3.19, there is a short list of low-degree polynomials δ-close to f_i, at least one of which is consistent with the PIOP. An out-of-domain query is used to force the prover to commit to a specific polynomial from this list.

- In conclusion, every sent polynomial oracle $\hat{f}_i$ is converted into a vector oracle f_i and the correctness of this vector, along with evaluations of $\hat{f}_i$, will be checked using an IOPP (WHIR).

4.5 Sumcheck

In order to understand WHIR, we first need to discuss the famous protocol for multilinear polynomials called **sumcheck**. Sumcheck was originally introduced in [LFKN92] but its prover efficiency was massively improved upon in [Tha13].

Sumcheck is a multi-round interactive oracle protocol that allows the prover $\mathcal{P}$ to prove to the verifier $\mathcal{V}$ that the sum of a multivariate polynomial $\hat{w} \in \mathbb{F}^{<d}[X_1, \ldots, X_m]$ over the Boolean hypercube is some value $\sigma \in \mathbb{F}$, i.e. that

$$\sum_{(b_1,\ldots,b_m)\in\mathbb{B}^m} \hat{w}(b_1, \ldots, b_m) = \sigma.$$

$\mathcal{V}$ has oracle access to $\hat{w}$.

Sumcheck is very verifier-efficient: $\mathcal{V}$ has to do $O(\log d) = O(m)$ field operations and only one oracle query to $\hat{w}$. If the protocol is optimized, $\mathcal{P}$ has to do $O(d) = O(2^m)$ work.

We give an overview of sumcheck in a non-optimized state.

- In the first round, the prover defines the *univariate* polynomial

  $$\hat{h}_1(X) := \sum_{(b_1,\ldots,b_{m-1})\in\mathbb{B}^{m-1}} \hat{w}(X, b_1, \ldots, b_{m-1})$$

  and sends it to the verifier (directly, not as an oracle). The verifier can check the initial sum $\sum_{\boldsymbol{b}\in\mathbb{B}^m} \hat{w}(\boldsymbol{b}) = \sigma$ by checking that $\hat{h}_1(0) + \hat{h}_1(1) = \sigma$. Thus, checking the original claim is reduced to checking the correctness of $\hat{h}_1$.

  $\mathcal{V}$ samples $\alpha_1 \leftarrow\!\$\ \mathbb{F}$ and sends it to the prover.

- In the second round, the prover uses α_1 to define a new *univariate* polynomial:

  $$\hat{h}_2(X) := \sum_{(b_1,\ldots,b_{m-2})\in\mathbb{B}^{m-2}} \hat{w}(\alpha_1, X, b_1, \ldots, b_{m-2}).$$

  $\mathcal{P}$ then sends this to $\mathcal{V}$ who checks that

  $$\hat{h}_2(0) + \hat{h}_2(1) = \hat{h}_1(\alpha_1),$$

  reducing checking the correctness of $\hat{h}_1$ to checking the correctness of $\hat{h}_2$.

$\mathcal{V}$ samples $\alpha_2 \leftarrow\!\!\$\ \mathbb{F}$ and sends it to $\mathcal{P}$.

- The protocol follows analogously: in iteration i, $\mathcal{P}$ defines

$$\hat{h}_i(X) := \sum_{(b_1,\ldots,b_{m-i})\in\mathbb{B}^{m-i}} \hat{w}(\alpha_1,\ldots,\alpha_{i-1},X,b_1,\ldots,b_{m-i}),$$

  $\mathcal{V}$ checks that $\hat{h}_i(0)+\hat{h}_i(1)=\hat{h}_{i-1}(\alpha_{i-1})$ and samples $\alpha_i \leftarrow\!\!\$\ \mathbb{F}$.

- In the iteration $i=m$, $\mathcal{V}$ has the claimed univariate polynomial

$$\hat{h}_m(X) = \hat{w}(\alpha_1,\ldots,\alpha_{m-1},X)$$

  and α_m. She checks consistency with $\hat{h}_{m-1}$ by checking that

$$\hat{h}_m(0)+\hat{h}_m(1)=\hat{h}_{m-1}(\alpha_{m-1}).$$

  She then makes her only query to the oracle for $\hat{w}(\alpha_1,\ldots,\alpha_m)$ and checks that

$$\hat{h}_m(\alpha_m)=\hat{w}(\alpha_1,\ldots,\alpha_m).$$

The core idea of sumcheck is that at each stage, the problem is simplified to have fewer variables. Verifier's check in iteration i shows that $\hat{h}_i$ is consistent with $\hat{h}_{i-1}$. The final query to the oracle is used to check that $\hat{h}_m$ is consistent with $\hat{w}$. This implies correctness of $\hat{h}_{m-1}$ which in turn implies the correctness of $\hat{h}_{m-2}$, etc, ending with the original claim.

The sumcheck protocol was shown to be sound in [LFKN92].

WHIR uses a slightly modified version of sumcheck since the function being summed over is not a polynomial $\hat{w}(X_1,\ldots,X_m)$, but a composition $\hat{w}(\hat{f}(X_1,\ldots,X_m),X_1,\ldots,X_m)$.

5 WHIR

In this chapter, we present the WHIR protocol as it was introduced in Chapter 5 of [ACFY25]. WHIR is based on ideas taken from [BBHR18], [ACFY24] and [ZCF24]. We first give an intuitive explanation to the protocol, which notably was absent in [ACFY25]. Afterwards we present WHIR in detail and analyze its parameters. We then prove its completeness and follow [ACFY25] in the proof of its round-by-round soundness. We finish the chapter with a worked example of using WHIR.

5.1 High level overview of WHIR

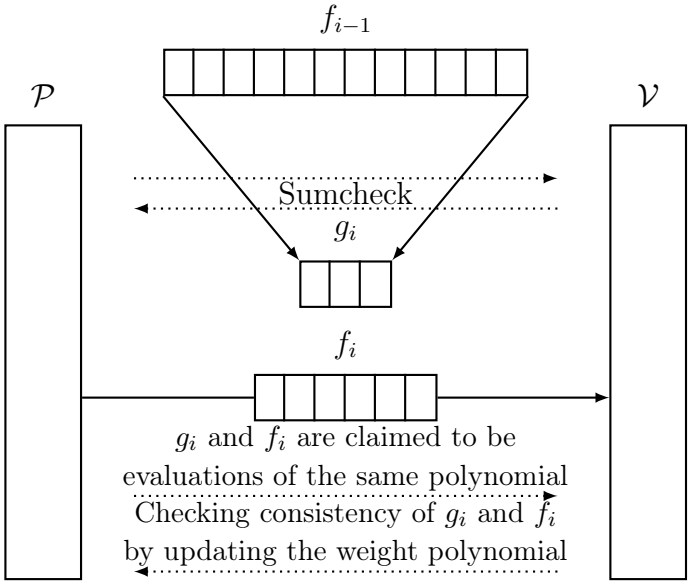

Figure 10: High level overview of one WHIR iteration.

0. **Setup**. The verifier $\mathcal{V}$ has oracle access to $f_0 \colon \mathcal{L} \to \mathbb{F}$. The goal of the protocol is for $\mathcal{P}$ to convince $\mathcal{V}$ that $\Delta(f_0, \mathcal{C}) \leq \delta$ for $\mathcal{C} := \mathrm{CRS}[\mathbb{F}, \mathcal{L}_0, m_0, \hat{w}_0, \sigma_0]$ and a chosen proximity parameter $\delta \in [0, 1]$.

   The honest prover $\mathcal{P}$ starts with a polynomial $\hat{f}_0 \in \mathbb{F}^{<2}[X_1, \ldots, X_m]$ that satisfies

$$\sum_{\boldsymbol{b} \in \mathbb{B}} \hat{w}_0(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sigma_0.$$

Then $f_0 = \hat{f}_0|_{\mathcal{L}_0}$.

WHIR has M iterations, each consisting of a domain shift followed by a sumcheck.

1. **Initial sumcheck**. The protocol begins with an initial sumcheck. The prover and verifier use sumcheck with randomness sampled by $\mathcal{V}$ to fix k_0 coordinates of $\hat{f}_0$. By Proposition 3.25, this yields the function

$$\mathrm{Fold}(f_0, (\alpha_{0,1}, \ldots, \alpha_{0,k_0})) \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_0^{(2^{k_0})}, m_0 - k_0].$$

The constraint $\sum_{\boldsymbol{b} \in \mathbb{B}} \hat{w}_0(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sigma_0$ is also reduced to a new constraint $\hat{h}_{0,k_0}(0) + \hat{h}_{0,k_0}(1) = \hat{h}_{0,k_0-1}(\alpha_{0,k_0-1})$.

2. **Main loop**. Now follow $M - 1$ iterations of interleaved sumchecks and domain shifts. We define $m_i := m_0 - \sum_{j=0}^{i-1} k_j$. Then m_i is the number of variables left after $i - 1$ sumchecks (the sumcheck at iteration i folds away k_i variables).

At iteration $i \in [M - 1]$, we define

$$\hat{f}_i := \hat{f}_{i-1}(\alpha_{i-1,1}, \ldots, \alpha_{i-1,k_{i-1}}, \cdot) = \hat{f}_0(\alpha_{0,1}, \ldots, \alpha_{i-1,k_{i-1}}, \cdot).$$

In other words, $\hat{f}_i \in \mathbb{F}^{<2}[X_1, \ldots, X_{m_i}]$ is the polynomial we get from $\hat{f}_0$ after all the folded (fixed) variables from previous sumchecks.

Since $\hat{f}_i = \mathrm{Fold}(\hat{f}_{i-1}, (\alpha_{i-1,1}, \ldots, \alpha_{i-1,k_{i-1}}))$, the natural definition would be $f_i = g_i := \mathrm{Fold}(f_{i-1}, (\alpha_{i-1,1}, \ldots, \alpha_{i-1,k_{i-1}}))$. As $g_i \colon \mathcal{L}_{i-1}^{(2^{k_{i-1}})} \to \mathbb{F}$, in this case we would have equality between code rates in each iteration:

$$\rho_i = \frac{2^{m_i}}{|\mathcal{L}_i|} = \frac{2^{m_{i-1}-k_{i-1}}}{|\mathcal{L}_{i-1}|/2^{k_{i-1}}} = \frac{2^{m_{i-1}}}{|\mathcal{L}_{i-1}|} = \rho_{i-1}.$$

It was shown in [ACFY24] that this is not optimal. This is because a lower code rate means more redundancy in the code word and this makes it easier to detect cheating for the verifier. Thus, having a lower code rate means that the verifier has to make fewer queries to the oracle to achieve the desired security threshold. We therefore define f_i over a larger domain than $|\mathcal{L}_{i-1}^{(2^{k_{i-1}})}|$ — in this way the code rate decreases each iteration.

Thus, we instead define $f_i \colon \mathcal{L}_i \to \mathbb{F}$ with $|\mathcal{L}_i| > |\mathcal{L}_{i-1}^{(2^{k_{i-1}})}|$ (usually, $|\mathcal{L}_i| = \frac{|\mathcal{L}_{i-1}|}{2}$). In this case,

$$\rho_i = \frac{2^{m_i}}{|\mathcal{L}_i|} < \frac{2^{m_{i-1}}}{|\mathcal{L}_{i-1}|} = \rho_{i-1}.$$

The prover sends the verifier an oracle to this f_i. But since the prover could have cheated, the verifier has to check that f_i does indeed correspond to $\hat{f}_i$, i.e., that $f_i = \hat{f}_i|_{\mathcal{L}_i}$.

The verifier can calculate values of g_i using the oracle to f_{i-1} and repeated use of the folding formula (4). She can also evaluate the multilinear extension of f_i (which is supposed to equal $\hat{f}_i$) at the same points using Proposition 2.4. But since calculating the multilinear extension requires calculating a large sum over a hypercube, instead of doing this directly, this check is incorporated into the weight polynomial $\hat{w}_i$.

In more detail, the verifier samples t_i values $z_1, \ldots, z_{t_i} \in \mathcal{L}_{i-1}^{(2^{k_{i-1}})}$, queries the oracle for f_{i-1} at the sets $\text{Block}(\mathcal{L}_{i-1}, k_{i-1}, z_{i,j})$, $j \in [t_i]$, uses the results to calculate $g_i(z_{i,j})$ and batches these consistency checks with f_i into $\hat{w}_i$. This step is called shifting the domain.

There remains only one problem. Suppose the prover had cheated and g_i was δ-far from all polynomials that satisfy these constraints. If we are not in the unique decoding regime (i.e. if $\delta > \mu/2$), there might be multiple code words of $\text{RS}[\mathbb{F}, \mathcal{L}_i, m_i]$ that are δ-close to f_i. The prover can then adaptively choose which one to use, based on the verifier queries. To overcome this, the prover is first (before the other queries) asked to evaluate $\hat{f}_i$ at a randomly sampled point $z_{i,0}$ and send $y_{i,0} := \hat{f}(z_{i,0})$ to the verifier. The verifier then batches the check that $\hat{f}_i(z_{i,0}) = y_{i,0}$ to $\hat{w}_i$.

If there were multiple polynomials close to f_i before, this single query forces the prover to commit to a concrete one of them because the probability that two different low degree polynomials that are δ-close to f_i agree at the point $z_{i,0}$ is negligible by Corollary 2.7.

After this, the prover and verifier go through a new sumcheck to fold away k_i more variables of f_0, thereby defining g_{i+1}.

3. **Send final polynomial**. After the main loop, m_M variables of f_0 remain

unfixed with m_M a small number. The resulting polynomial $\hat{f}_M$ is small enough to be sent directly to the verifier (without an oracle).

4. **Sample final randomness**. The verifier finishes by sampling t_M last elements from $\mathcal{L}_{M-1}^{\binom{2^{k_{M-1}}}{}}$ which she uses to check correctness of $\hat{f}_M$.

5.2 WHIR construction

Construction 5.1 (WHIR. [ACFY25] Construction 5.1)**.** [4]

- **Ingredients and notation.**
  - $\mathcal{C}$ is a constrained Reed-Solomon code with m_0 variables, weight polynomial $\hat{w}_0$ and target $\sigma_0 \in \mathbb{F}$ over a smooth evaluation domain $\mathcal{L}_0$, i.e. $\mathcal{C} = \mathrm{CRS}[\mathbb{F}, \mathcal{L}_0, m_0, \hat{w}_0, \sigma_0]$;
  - $M \in \mathbb{N}$ is the iteration count;
  - $k_0, \ldots, k_{M-1} \in \mathbb{N}$ are folding parameters that satisfy $\sum_{i=0}^{M-1} k_i \leq m_0$;
  - $m_i := m_0 - \sum_{j=0}^{i-1} k_j$ for $i \in [M]$;
  - $\mathcal{L}_1, \ldots, \mathcal{L}_{M-1} \subseteq \mathbb{F}$ are smooth evaluation domains with order $|\mathcal{L}_i| \geq 2^{m_i}$ (typically, $|\mathcal{L}_i| = \frac{|\mathcal{L}|}{2^i}$);
  - $t_1, \ldots, t_M \in \mathbb{N}$ are repetition parameters with $t_i \leq |\mathcal{L}_{i-1}|$;
  - $d^* := 1 + \deg_Z(\hat{w}_0) + \max_{i \in [m_0]} \deg_{X_i}(\hat{w}_0)$ and $d := \max\{d^*, 3\}$.

We now describe what the protocol looks like when both $\mathcal{P}$ and $\mathcal{V}$ are honest.

- **Inputs.** $\mathcal{V}$ has oracle access to $f_0 \colon \mathcal{L}_0 \to \mathbb{F}$, where $f_0 \in \mathcal{C}$. $\mathcal{P}$ receives $\hat{f}_0 \in \mathbb{F}^{<2}[X_1, \ldots, X_{m_0}]$ such that f_0 is $\hat{f}_0$ restricted to $\mathcal{L}_0$ and $\sum_{\boldsymbol{b} \in \mathbb{B}^{m_0}} \hat{w}_0(\hat{f}_0(\boldsymbol{b}), \boldsymbol{b}) = \sigma_0$.

- **Interactive phase.**
  1. **Initial sumcheck**. Set $\boldsymbol{\alpha}_0 = \emptyset$. For $\ell = 1, \ldots, k_0$:

[4] Changed inconsistent indexing of t_i.

(a) $\mathcal{P}$ sends $\mathcal{V}$ the univariate polynomial $\hat{h}_{0,\ell}(X) \in \mathbb{F}^{<d^*}[X]$:

$$\hat{h}_{0,\ell}(X) := \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0 - \ell}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_0, X, \boldsymbol{b}), \boldsymbol{\alpha}_0, X, \boldsymbol{b}).$$

(b) $\mathcal{V}$ samples $\alpha_{0,\ell} \leftarrow\!\!\$\ \mathbb{F}$ and sends it to $\mathcal{P}$. They update $\boldsymbol{\alpha}_0 := \boldsymbol{\alpha}_0 \| \alpha_{0,\ell}$.

2. **Main loop.** For each $i = 1, \ldots, M-1$:

(a) **Send folded function:** $\mathcal{P}$ sends $f_i \colon \mathcal{L}_i \to \mathbb{F}$ to an oracle which $\mathcal{V}$ can query, where f_i is the evaluation of $\hat{f}_i := \hat{f}_{i-1}(\boldsymbol{\alpha}_{i-1}, \cdot)$ over $\mathcal{L}_i$.

(b) **Out-of-domain sample:** $\mathcal{V}$ sends $z_{i,0} \leftarrow\!\!\$\ \mathbb{F}$ to $\mathcal{P}$ and they set $\boldsymbol{z}_{i,0} := \mathrm{pow}(z_{i,0}, m_i)$.

(c) **Out-of-domain reply:** $\mathcal{P}$ sends $y_{i,0} := \hat{f}_i(\boldsymbol{z}_{i,0})$ to $\mathcal{V}$.

(d) **Shift message:** $\mathcal{V}$ samples $z_{i,1}, \ldots, z_{i,t_i} \leftarrow\!\!\$\ \mathcal{L}_{i-1}^{\binom{2^{k_{i-1}}}{}}$ and $\gamma_i \leftarrow\!\!\$\ \mathbb{F}$ and sends them to $\mathcal{P}$. They set $\boldsymbol{z}_{i,j} := \mathrm{pow}(z_{i,j}, m_i)$.

(e) **Sumcheck:** Set $\boldsymbol{\alpha}_i = \emptyset$. For $\ell = 1, \ldots, k_i$:

i. $\mathcal{P}$ sends $\mathcal{V}$ the univariate polynomial $\hat{h}_{i,\ell} \in \mathbb{F}^{<d}[X]$, where

$$\hat{h}_{i,\ell}(X) := \sum_{\boldsymbol{b} \in \mathbb{B}^{m_i - \ell}} \hat{w}_i(\hat{f}_i(\boldsymbol{\alpha}_i, X, \boldsymbol{b}), \boldsymbol{\alpha}_i, X, \boldsymbol{b}),$$

where

$$\hat{w}_i(Z, X_1, \ldots, X_{m_i}) := \hat{w}_{i-1}(Z, \boldsymbol{\alpha}_{i-1}, X_1, \ldots, X_{m_i}) + \sum_{j=0}^{t_i} \gamma_i^{j+1} Z\, \mathsf{eq}(\boldsymbol{z}_{i,j}, (X_1, \ldots, X_{m_i})).$$

ii. The verifier samples $\alpha_{i,\ell} \leftarrow\!\!\$\ \mathbb{F}$ and sends it to $\mathcal{P}$. They update $\boldsymbol{\alpha}_i := \boldsymbol{\alpha}_i \| \alpha_{i,\ell}$.

3. **Send final polynomial:** $\mathcal{P}$ sends $\hat{f}_M \in \mathbb{F}^{<2}[X_1, \ldots, X_{m_M}]$ to $\mathcal{V}$, where $\hat{f}_M := \hat{f}_{M-1}(\boldsymbol{\alpha}_{M-1}, \cdot)$.

4. **Sample final randomness:** $\mathcal{V}$ samples $r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}} \leftarrow\!\!\$\ \mathcal{L}_{M-1}^{\binom{2^{k_{M-1}}}{}}$.

- **Decision phase.** $\mathcal{V}$ carries out the following checks.

1. **Check initial sumcheck:**
   (a) Check that $\hat{h}_{0,1}(0) + \hat{h}_{0,1}(1) = \sigma_0$.
   (b) Check that
   $$\hat{h}_{0,\ell}(0) + \hat{h}_{0,\ell}(1) = \hat{h}_{0,\ell-1}(\alpha_{0,\ell-1})$$
   for every $\ell \in \{2, \dots, k_0\}$.
2. **Check main loop:** For $i = 1, \dots, M-1$:
   (a) Let $g_{i-1} := \text{Fold}(f_{i-1}, \boldsymbol{\alpha}_{i-1})$.
   (b) Compute the points $\{g_{i-1}(z_{i,j})\}_{j \in [t_i]}$ by querying the oracle for evaluations of f_{i-1} at the points in $\text{Block}(\mathcal{L}_{i-1}, k_{i-1}, z_{i,j})$.
   (c) Check that
   $$\hat{h}_{i,1}(0) + \hat{h}_{i,1}(1) = \hat{h}_{i-1,k_{i-1}}(\alpha_{i-1,k_{i-1}}) + \gamma_i y_{i,0} + \sum_{j=1}^{t_i} \gamma_i^{j+1} g_{i-1}(z_{i,j}).$$
   (d) Check that
   $$\hat{h}_{i,\ell}(0) + \hat{h}_{i,\ell}(1) = \hat{h}_{i,\ell-1}(\alpha_{i,\ell-1})$$
   for every $\ell \in \{2, \dots, k_i\}$.
3. **Check final polynomial:**
   (a) Compute the points $\{g_{M-1}(r_\ell^{\text{fin}})\}_{\ell \in [t_M]}$ by querying the oracle for evaluations of f_{M-1} at the points in $\text{Block}(\mathcal{L}_{M-1}, k_{M-1}, r_\ell^{\text{fin}})$.
   (b) Check that
   $$\hat{f}_M(\boldsymbol{r}_\ell^{\text{fin}}) = g_{M-1}(r_\ell^{\text{fin}})$$
   for every $\ell \in [t_M]$, where $\boldsymbol{r}_\ell^{\text{fin}} = \text{pow}(r_\ell^{\text{fin}}, m_M)$.
   (c) Check that
   $$\sum_{\boldsymbol{b} \in \mathbb{B}^{m_M}} \hat{w}_{M-1}(\hat{f}_M(\boldsymbol{b}), \boldsymbol{\alpha}_{M-1}, \boldsymbol{b}) = \hat{h}_{M-1,k_{M-1}}(\alpha_{M-1,k_{M-1}}).$$

The protocol can be seen on Figure 11. Later, in Example 5.8, we apply WHIR on a concrete problem.

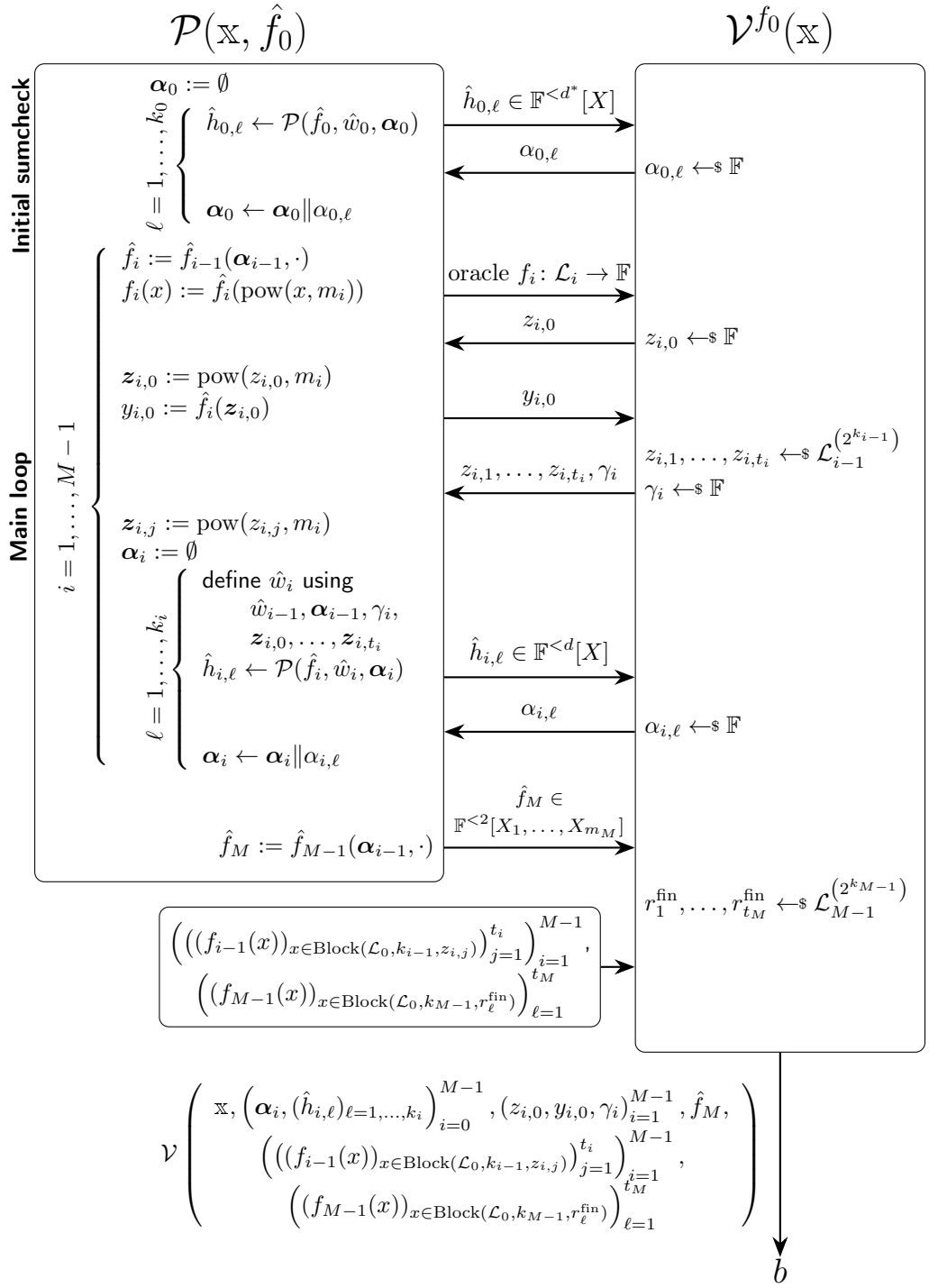

Figure 11: WHIR IOPP between prover $\mathcal{P}$ and verifier $\mathcal{V}$. Here, $\mathbb{x} = (\mathcal{C}, M, k_0, \ldots, k_{M-1}, \mathcal{L}_1, \ldots, \mathcal{L}_{M-1}, t_1, \ldots, t_M)$.

5.2.1 WHIR parameters

We analyze the parameters of Construction 5.1.

- **Rounds**[5] . The protocol has

$$k_0 + 2(M-1) + \left(\sum_{i=1}^{M-1} k_i\right) + 1 = O\left(M + \sum_{i=0}^{M-1} k_i\right)$$

  rounds.

- **Proof length**.

  1. In the initial sumcheck, the prover sends k_0 degree $< d^*$ polynomials, corresponding to sending $k_0 d^*$ field elements.
  2. In the i-th iteration of the main loop, the prover sends an oracle message of length $|\mathcal{L}_i|$, a field element, and k_i polynomials of degree $< d$. This corresponds to $|\mathcal{L}_i| + 1 + k_i d$ field elements.
  3. At the end of the protocol, the prover sends one multilinear polynomial of m_M variables, corresponding to 2^{m_M} field elements.

  Altogether, the proof length is

$$k_0 d^* + \sum_{i=1}^{M-1} (|\mathcal{L}_i| + 1 + k_i d) + 2^{m_M} = O\left(\sum_{i=1}^{M-1} (|\mathcal{L}_i| + k_i)\right).$$

  Since m_M is a small constant, $\sum_{i=1}^{M-1} k_i = m - m_M = O(m)$. In usual applications $|\mathcal{L}_i| = |\mathcal{L}|/2^i = n2^{-i}$, giving the proof length

$$O(Mn + m_0).$$

- **Query complexity**. For each $i \in [M]$, the verifier queries the oracle for f_{i-1} at $|\operatorname{Block}(\mathcal{L}_{i-1}, k_{i-1}, z_{i,j})| = 2^{k_{i-1}}$ points t_i times. But since the $2^{k_{i-1}}$ points in $\operatorname{Block}(\mathcal{L}_{i-1}, k_{i-1}, z_{i,j})$ are always queried together, they can be grouped into a single symbol. This means that the query complexity is $\sum_{i=1}^{M} t_i$.

[5] This is incorrect in [ACFY25].

- **Verifier sampling.** The verifier has to sample $\sum_{i=0}^{M-1} k_i + M - 1$ field elements and t_i elements from $\mathcal{L}_{i-1}^{\binom{2^{k_{i-1}}}{}}$ for each $i \in [M]$.

- **Verifier work.**[6] We know that $d = \max\{3, d^*\} = \Theta(d^*)$ so we write everything in terms of d^*. We know that $M \leq \sum_{i=1}^{M-1} k_i$. Finally, we also assume that m_M is very small.

  1. **Check initial sumcheck.** The verifier has to evaluate the degree $< d^*$ univariate polynomials $\hat{h}_{0,1}, \ldots, \hat{h}_{0,k_0}$ at 0 and 1. Evaluating at 0 corresponds to reading out the free term and evaluating at 1 corresponds to doing up to $d^* - 1$ additions. She also has to evaluate $\hat{h}_{0,1}, \ldots, \hat{h}_{0,k_0}$ at a random field element. This takes $O(k_0 d^*)$ field operations.

  2. **Check main loop.** For $i \in [M-1]$:

     (a) To compute the points $\{g_{i-1}(z_{i,j})\}_{j \in [t_i]}$, the verifier needs to fold t_i points with depth k_{i-1}. Each individual folding takes $O(1)$ field operations and since we fold in a binary tree, it takes $2^{k_{i-1}}$ foldings for one point. Altogether, this step takes $O(t_i 2^{k_{i-1}})$ operations.

     (b) Checking

     $$\hat{h}_{i,1}(0) + \hat{h}_{i,1}(1) = \hat{h}_{i-1,k_{i-1}}(\alpha_{i-1,k_{i-1}}) + \gamma_i y_{i,0} + \sum_{j=1}^{t_i} \gamma_i^{j+1} g_{i-1}(z_{i,j})$$

     takes $O(d^* + t_i)$ field operations.

     (c) Checking

     $$\hat{h}_{i,\ell}(0) + \hat{h}_{i,\ell}(1) = \hat{h}_{i,\ell-1}(\alpha_{i,\ell-1})$$

     for each $\ell \in [k_i]$ takes $O(k_i d^*)$ field operations.

     Altogether, this phase takes

     $$O\left(\sum_{i=1}^{M-1} \left(t_i 2^{k_{i-1}} + k_i d^*\right)\right)$$

[6] Analysis of verifier work is only done for a simplified version of WHIR in [ACFY25].

field operations (we used $M \leq \sum_{i=1}^{M-1} k_i$).

3. **Check final polynomial.**

   (a) Folding takes $O(t_M 2^{k_{M-1}})$ operations.

   (b) Evaluating a multilinear polynomial in m_M variables at a point takes $O(m_M) = O(1)$ operations. So evaluating $\hat{f}_M$ at t_M points takes $O(t_M)$ operations.

   (c) To check

$$\sum_{\boldsymbol{b} \in \mathbb{B}^{m_M}} \hat{w}_{M-1}(\hat{f}_M(\boldsymbol{b}), \boldsymbol{\alpha}_{M-1}, \boldsymbol{b}) = \hat{h}_{M-1,k_{M-1}}(\alpha_{M-1,k_{M-1}}),$$

   the verifier needs to evaluate $\hat{w}_{M-1}$ at $2^{m_M} = O(1)$ points. Evaluating $\hat{w}_{M-1}$ at a single point costs $O(d^* + M)$ field operations: $O(d^*)$ to evaluate the low-degree part inherited from $\hat{w}_0$, and $O(M)$ to evaluate the $O(M)$ equality polynomials, each of which depends on only $m_M = O(1)$ variables. Evaluating $\hat{h}_{M-1,k_{M-1}}(\alpha_{M-1,k_{M-1}})$ takes an additional $O(d^*)$ field operations. This step takes $O(d^* + M)$ field operations in total.

   Adding it all together, the whole verifier work simplifies to

$$O\left(d^* \sum_{i=0}^{M-1} k_i + \sum_{i=1}^{M} t_i 2^{k_{i-1}}\right) = O\left(d^* m_0 + \sum_{i=1}^{M} t_i 2^{k_{i-1}}\right).$$

   We show later in Chapter 5.4.4 that to achieve λ-bit security, we must take

$$t_i = O\left(\frac{\lambda}{\log_2(1/\rho_{i-1})}\right).$$

5.3 WHIR is complete

In this chapter, we show that Construction 5.1 is complete (cf. Definition 4.5). This is not done in [ACFY25].

Proposition 5.2. *The WHIR protocol 5.1 is complete.*

Proof. We assume that $\mathcal{P}$ and $\mathcal{V}$ are honest and go through the decision stage of the protocol. We define $\boldsymbol{\alpha}_{i,\ell} := (\alpha_{i,1}, \ldots, \alpha_{i,\ell})$ for all $\ell \in \{1, \ldots, k_i\}$ and $i \in$

$\{0, \ldots, M-1\}$.

1. **Check initial sumcheck.**

   (a) By definition of $\hat{h}_{0,1}$, we have

   $$\begin{aligned}\hat{h}_{0,1}(0) + \hat{h}_{0,1}(1) &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-1}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_0, 0, \boldsymbol{b}), \boldsymbol{\alpha}_0, 0, \boldsymbol{b}) \\ &\quad + \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-1}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_0, 1, \boldsymbol{b}), \boldsymbol{\alpha}_0, 1, \boldsymbol{b}) \\ &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0}} \hat{w}_0(\hat{f}_0(\boldsymbol{b}), \boldsymbol{b}) \\ &= \sigma_0.\end{aligned}$$

   (b) Let $\ell \in \{2, \ldots, k_0\}$. In this stage of the protocol, $\boldsymbol{\alpha}_0 = \boldsymbol{\alpha}_{0,\ell-1}$. We have

   $$\begin{aligned}\hat{h}_{0,\ell}(0) + \hat{h}_{0,\ell}(1) &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-\ell}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_0, 0, \boldsymbol{b}), \boldsymbol{\alpha}_0, 0, \boldsymbol{b}) \\ &\quad + \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-\ell}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_0, 1, \boldsymbol{b}), \boldsymbol{\alpha}_0, 1, \boldsymbol{b}) \\ &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-\ell+1}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_{0,\ell-1}, \boldsymbol{b}), \boldsymbol{\alpha}_{0,\ell-1}, \boldsymbol{b}) \\ &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_0-(\ell-1)}} \hat{w}_0(\hat{f}_0(\boldsymbol{\alpha}_{0,\ell-2}, \alpha_{0,\ell-1}, \boldsymbol{b}), \boldsymbol{\alpha}_{0,\ell-2}, \alpha_{0,\ell-1}, \boldsymbol{b}) \\ &= \hat{h}_{0,\ell-1}(\alpha_{0,\ell-1}).\end{aligned}$$

2. **Check main loop.** Let $i \in \{1, \ldots, M-1\}$.

   (a) Let $g_{i-1} := \mathrm{Fold}(f_{i-1}, \boldsymbol{\alpha}_{i-1})$, $\boldsymbol{z}_{i,j} = \mathrm{pow}(z_{i,j}, m_i)$, $\hat{f}_i := \hat{f}_{i-1}(\boldsymbol{\alpha}_{i-1}, \cdot)$, and $y_{i,0} := \hat{f}_i(\boldsymbol{z}_{i,0})$. Since $m_i = m_0 - \sum_{j=0}^{i-1} k_j$, we have $m_i = m_{i-1} - k_{i-1}$. As the iteration $i-1$ is over, we have $\boldsymbol{\alpha}_{i-1} = \boldsymbol{\alpha}_{i-1,k_{i-1}}$.

(c) At the beginning of the i-th iteration, $\boldsymbol{\alpha}_i = \emptyset$. Thus,

$$\begin{aligned}
\hat{h}_{i,1}(0) + \hat{h}_{i,1}(1) &= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-1}} \hat{w}_i(\hat{f}_i(0,\boldsymbol{b}),0,\boldsymbol{b}) + \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-1}} \hat{w}_i(\hat{f}_i(1,\boldsymbol{b}),1,\boldsymbol{b}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{w}_i(\hat{f}_i(\boldsymbol{b}),\boldsymbol{b}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \Bigg(\hat{w}_{i-1}(\hat{f}_i(\boldsymbol{b}),\boldsymbol{\alpha}_{i-1},\boldsymbol{b}) \\
&\quad + \sum_{j=0}^{t_i} \gamma_i^{j+1} \hat{f}_i(\boldsymbol{b})\,\mathsf{eq}(\boldsymbol{z}_{i,j},\boldsymbol{b}) \Bigg) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_{i-1}-k_{i-1}}} \hat{w}_{i-1}\big(\hat{f}_{i-1}(\boldsymbol{\alpha}_{i-1,k_{i-1}-1},\alpha_{i-1,k_{i-1}},\boldsymbol{b}), \\
&\qquad\qquad \boldsymbol{\alpha}_{i-1,k_{i-1}-1},\alpha_{i-1,k_{i-1}},\boldsymbol{b}\big) \\
&\quad + \gamma_i \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{f}_i(\boldsymbol{b})\,\mathsf{eq}(\boldsymbol{z}_{i,0},\boldsymbol{b}) \\
&\quad + \sum_{j=1}^{t_i} \gamma_i^{j+1} \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{f}_{i-1}(\boldsymbol{\alpha}_{i-1},\boldsymbol{b})\,\mathsf{eq}(\boldsymbol{z}_{i,j},\boldsymbol{b}) \\
&= \hat{h}_{i-1,k_{i-1}}(\alpha_{i-1,k_{i-1}}) + \gamma_i y_{i,0} + \sum_{j=1}^{t_i} \gamma_i^{j+1} g_{i-1}(z_{i,j})
\end{aligned}$$

where the last equality holds based on Definition 2.5 and Proposition 3.25.

(d) Let $\ell \in \{2,\ldots,k_i\}$. Then $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i,\ell-1}$ and

$$\begin{aligned}
\hat{h}_{i,\ell}(0) + \hat{h}_{i,\ell}(1) &= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-\ell}} \hat{w}_i(\hat{f}_i(\boldsymbol{\alpha}_{i,\ell-1},0,\boldsymbol{b}),\boldsymbol{\alpha}_{i,\ell-1},0,\boldsymbol{b}) \\
&\quad + \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-\ell}} \hat{w}_i(\hat{f}_i(\boldsymbol{\alpha}_{i,\ell-1},1,\boldsymbol{b}),\boldsymbol{\alpha}_{i,\ell-1},1,\boldsymbol{b}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-\ell+1}} \hat{w}_i(\hat{f}_i(\boldsymbol{\alpha}_{i,\ell-1},\boldsymbol{b}),\boldsymbol{\alpha}_{i,\ell-1},\boldsymbol{b}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}^{m_i-(\ell-1)}} \hat{w}_i(\hat{f}_{i-1}(\boldsymbol{\alpha}_{i,\ell-2},\alpha_{i,\ell-1},\boldsymbol{b}),\boldsymbol{\alpha}_{i,\ell-2},\alpha_{i,\ell-1},\boldsymbol{b}) \\
&= \hat{h}_{i,\ell-1}(\alpha_{i,\ell-1}).
\end{aligned}$$

3. **Check final polynomial.**

(a) Let $\ell \in [t_M]$ and $\boldsymbol{r}_\ell^{\text{fin}} = \text{pow}(r_\ell^{\text{fin}}, m_M)$. We have

$$\begin{aligned}\hat{f}_M(\boldsymbol{r}_\ell^{\text{fin}}) &= \hat{f}_{M-1}(\boldsymbol{\alpha}_{M-1}, \boldsymbol{r}_\ell^{\text{fin}}) \\ &= \text{Fold}(f_{M-1}, \boldsymbol{\alpha}_{M-1})(r_\ell^{\text{fin}}) \\ &= g_{M-1}(r_\ell^{\text{fin}}).\end{aligned}$$

(b) We have $m_M = m_{M-1} - k_{M-1}$. Thus,

$$\begin{aligned}&\sum_{\boldsymbol{b} \in \mathbb{B}^{m_M}} \hat{w}_{M-1}(\hat{f}_M(\boldsymbol{b}), \boldsymbol{\alpha}_{M-1}, \boldsymbol{b}) \\ &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_{M-1}-k_{M-1}}} \hat{w}_{M-1}(\hat{f}_{M-1}(\boldsymbol{\alpha}_{M-1}, \boldsymbol{b}), \boldsymbol{\alpha}_{M-1}, \boldsymbol{b}) \\ &= \sum_{\boldsymbol{b} \in \mathbb{B}^{m_{M-1}-k_{M-1}}} \hat{w}_{M-1}\Big(\hat{f}_{M-1}(\boldsymbol{\alpha}_{M-1,k_{M-1}-1}, \alpha_{M-1,k_{M-1}}, \boldsymbol{b}), \\ &\qquad\qquad \boldsymbol{\alpha}_{M-1,k_{M-1}-1}, \alpha_{M-1,k_{M-1}}, \boldsymbol{b}\Big) \\ &= \hat{h}_{M-1,k_{M-1}}(\alpha_{M-1,k_{M-1}}).\end{aligned}$$

We have shown that if $\mathcal{P}$ and $\mathcal{V}$ are honest, all checks in the decision stage pass. This means that Construction 5.1 is complete. □

5.4 WHIR is round-by-round sound

In this chapter we show that Construction 5.1 is round-by-round sound (c.f. Definition 4.4), very closely following what is done in Chapter 5.1 in [ACFY25].

At the end of the chapter, we analyze choices of parameters for $\lambda = 128$.

Theorem 5.3 ([ACFY25] Theorem 5.2)**.** [7] *Consider Construction 5.1 with the same ingredients and notation. For every $f \notin \mathcal{C}$ and every $\delta_0, \ldots, \delta_{M-1}$ and $(\ell_{i,s})_{\substack{0 \le i \le M-1 \\ 0 \le s \le k_i}}$ where*

- $\delta_0 \in (0, \Delta(f, \mathcal{C}))$*;*
- *the function $\texttt{Gen}(\ell; \alpha) = (1, \alpha, \ldots, \alpha^{\ell-1})$ is a proximity generator with mutual correlated agreement with proximity bound $\texttt{B}^\star$ satisfying $\texttt{B}^\star(\mathcal{C}_{\text{RS}}^{(i,s)}, 2) \ge$*

[7] We introduce explicit definitions for some variables that are used but not formally defined in [ACFY25], correct minor typographical errors, and improve the exposition.

δ_i *and error* $\mathtt{err}^\star$ *for each of the codes* $(\mathcal{C}_{\mathrm{RS}}^{(i,s)})_{\substack{0\le i\le M-1\\ 0\le s\le k_i}}$ *where* $\mathcal{C}_{\mathrm{RS}}^{(i,s)} := \mathrm{RS}[\mathbb{F}, \mathcal{L}_i^{(2^s)}, m_i - s]$*;*

- *for every* $0 \le i \le M-1$*,* $\delta_i \in (0, 1 - \mathtt{B}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, 2))$*;*
- *for every* $0 \le i \le M-1$*,* $0 \le s \le k_i$*,* $\mathcal{C}_{\mathrm{RS}}^{(i,s)}$ *is* $(\ell_{i,s}, \delta_i)$*-list decodable.*

Then Construction 5.1 is an IOPP for $\mathcal{C}$ *with round-by-round soundness error*

$$((\varepsilon_{0,s}^{\mathrm{fold}})_{s\le k_0}, (\varepsilon_i^{\mathrm{out}}, \varepsilon_i^{\mathrm{shift}}, (\varepsilon_{i,s}^{\mathrm{fold}})_{s\le k_i})_{i\le M-1}, \varepsilon^{\mathrm{fin}})$$

where

- $\varepsilon_{0,s}^{\mathrm{fold}} \le \dfrac{d^* \ell_{0,s-1}}{|\mathbb{F}|} + \mathtt{err}^\star(\mathcal{C}_{\mathrm{RS}}^{(0,s)}, 2, \delta_0)$*;*
- $\varepsilon_i^{\mathrm{out}} \le \dfrac{2^{m_i} \ell_{i,0}^2}{2|\mathbb{F}|}$*;*
- $\varepsilon_i^{\mathrm{shift}} \le (1-\delta_{i-1})^{t_i} + \dfrac{\ell_{i,0}(t_i+1)}{|\mathbb{F}|}$*;*
- $\varepsilon_{i,s}^{\mathrm{fold}} \le \dfrac{d\ell_{i,s-1}}{|\mathbb{F}|} + \mathtt{err}^\star(\mathcal{C}_{\mathtt{RS}}^{(i,s)}, 2, \delta_i)$*;*
- $\varepsilon^{\mathrm{fin}} \le (1-\delta_{M-1})^{t_M}$*.*

We first provide some notation that will be used throughout this chapter, then define the state function for the protocol, then finally prove the error bounds in Theorem 5.3.

5.4.1 Notation

We will use the following notation.

- $\boldsymbol{\alpha}_{i,\ell} := (\alpha_{i,1}, \ldots, \alpha_{i,\ell})$.
- $f_{i,0} := f_i$ and $f_{i,\ell} = \mathrm{Fold}(f_i, \boldsymbol{\alpha}_{i,\ell}) = \mathrm{Fold}(f_{i,\ell-1}, \alpha_{i,\ell})$. Then $g_i = f_{i,k_i}$.
- $\hat{w}_{i,\ell}(Z, X_1, \ldots, X_{m_i-\ell}) := \hat{w}_i(Z, \boldsymbol{\alpha}_{i,\ell}, X_1, \ldots, X_{m_i-\ell})$.

- $\sigma_{0,0} := \sigma_0$,
  $\sigma_{i,0} := \hat{h}_{i-1,k_{i-1}}(\alpha_{i-1,k_{i-1}}) + \gamma_i y_{i,0} + \sum_{j=1}^{t_i} \gamma_i^{j+1} g_{i-1}(z_{i,j})$,
  $\sigma_{i,\ell} := \hat{h}_{i,\ell-1}(\alpha_{i,\ell-1})$ for $\ell \in [k_i]$ and $i \in \{0, \ldots, M-1\}$.
- $\mathcal{C}_{\mathrm{RS}}^{(i,s)} := \mathrm{RS}[\mathbb{F}, \mathcal{L}_i^{(2^s)}, m_i - s]$.
- $\mathcal{C}_{\mathrm{CRS}}^{(i,s)} := \mathrm{CRS}[\mathbb{F}, \mathcal{L}_i^{(2^s)}, m_i - s, \hat{w}_{i,\ell}, \sigma_{i,\ell}]$.

The full transcript of the protocol is

$$\mathtt{tr} = \left(\begin{array}{c} \left(\hat{h}_{0,\ell}, \alpha_{0,\ell}\right)_{\ell \le k_0}, \\ \left(f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \le k_i}\right)_{i \le M-1}, \\ \hat{f}_M, \\ r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}} \end{array} \right).$$

The first row above corresponds to the initial sumcheck, the second row to the main loop, the third row to sending the final polynomial and the last row to sampling final randomness in the description of Construction 5.1.

We say that a partial transcript is **sumcheck-valid** if it passes all the verifier's checks of the decision stage that can be derived from it in Item 1 and 2 of the decision stage.

5.4.2 The state function

We now define a state function `State` for the protocol (cf. Definition 4.3). Note that we have to define its value after every action taken by the verifier but not after prover's actions.

Formally, `State` also gets the initial inputs f_0, $\hat{w}_0$ and σ_0 as arguments but we omit them in the following and only consider the transcript. We also write in brackets after each point why we define it as such for intuition.

0. For the initial function f_0, we set $\mathtt{State}(\emptyset) = 1$ if and only if $f_0 \in \mathcal{C}$.
1. **Initial sumcheck, round** $s \in [k_0]$. At this point,
$$\mathtt{tr} = \left((\hat{h}_{0,\ell}, \alpha_{0,\ell})_{\ell \le s-1}, \hat{h}_{0,s}\right),$$

and $\mathcal{V}$ samples $\alpha_{0,s}$. We set $\texttt{State}(\texttt{tr}\|\alpha_{0,s}) = 1$ if and only if both the following hold.

(a) $(\texttt{tr}\|\alpha_{0,s})$ is sumcheck-valid.

(b) $|\operatorname{List}[\mathcal{C}_{\mathrm{CRS}}^{(0,s)}, f_{0,s}, \delta_0]| \geq 1$.

(If $\texttt{tr}$ is not sumcheck-valid then $\mathcal{V}$ will catch the cheating $\mathcal{P}$. If Item 1b is not the case, there is no code word δ_0-close to the s-times folded code. And by Theorem 3.41, this means that w.h.p. f_0 itself was not close to a code word.)

2. **Out-of-domain sample at iteration** i. At this point

$$\texttt{tr} = \left(\begin{array}{c} \left(\hat{h}_{0,\ell}, \alpha_{0,\ell}\right)_{\ell \leq k_0}, \\ \left(f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \leq k_i}\right)_{j \leq i-1}, \\ f_i \end{array} \right).$$

and $\mathcal{V}$ samples $z_{i,0} \leftarrow\!\!\$\ \mathbb{F}$. We set $\texttt{State}(\texttt{tr}\|z_{i,0}) = 1$ if and only if at least one of the following holds.

(a) The previous state function reverts, i.e. the following both hold:

i. $(\texttt{tr}\|z_{i,0})$ is sumcheck-valid;

ii. $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i-1,k_{i-1})}, g_{i-1}, \delta_{i-1})| \geq 1$.

(If the transcript was not sumcheck valid before, concatenating $z_{i,0}$ to $\texttt{tr}$ will not change this. This condition is to ensure that an honest prover keeps $\texttt{State}(\texttt{tr}\|z_{i,0}) = 1$.)

(b) There are multiple consistent code words: $\exists u, u' \in \operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i,0)}, f_i, \delta_i)$ for which $\hat{u}(\boldsymbol{z}_{i,0}) = \hat{u}'(\boldsymbol{z}_{i,0})$ where $\boldsymbol{z}_{i,0} = \operatorname{pow}(z_{i,0}, m_i)$. (We cannot tell if $\mathcal{P}$ is cheating if the out-of-domain query failed to constrain him to a single code word.)

3. **Shift message at iteration** i. At this point,

$$\texttt{tr} = \left(\begin{array}{c} \left(\hat{h}_{0,\ell}, \alpha_{0,\ell}\right)_{\ell \leq k_0}, \\ \left(f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \leq k_i}\right)_{j \leq i-1}, \\ f_i, z_{i,0}, y_{i,0} \end{array} \right).$$

The verifier samples $z_{i,1}, \ldots, z_{i,t_i}$ and γ_i and we set

$$\texttt{State}(\texttt{tr} \| (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i)) = 1$$

if and only if both the following hold:

(a) $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i,0)}, f_i, \delta_i)| \geq 1$;

(b) $\texttt{tr} \| (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i)$ is sumcheck-valid.

(There needs to be a code word close to the folded function f_i which satisfies the sumcheck constraints.)

4. **Sumcheck randomness at round $1 \leq s \leq k_i$ of iteration i.** At this point,

$$\texttt{tr} = \left(\begin{array}{c} \left(\hat{h}_{0,\ell}, \alpha_{0,\ell}\right)_{\ell \leq k_0}, \\ \left(f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \leq k_i}\right)_{j \leq i-1}, \\ f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \leq s-1}, \hat{h}_{i,s} \end{array} \right)$$

and the verifier samples $\alpha_{i,s}$. We set $\texttt{State}(\texttt{tr}) = 1$ if and only if both the following hold:

(a) $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i,s)}, f_{i,s}, \delta_i)| \geq 1$,

(b) $(\texttt{tr} \| \alpha_{i,s})$ is sumcheck-valid.

(If there is no code word δ_i-close to $f_{i,s}$ then w.h.p. f_i was not close to a code word. If sumcheck-validity is violated then the prover will certainly be caught.)

5. **Final randomness.** At this point,

$$\texttt{tr} = \left(\begin{array}{c} \left(\hat{h}_{0,\ell}, \alpha_{0,\ell}\right)_{\ell \leq k_0}, \\ \left(f_i, z_{i,0}, y_{i,0}, (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i), (\hat{h}_{i,\ell}, \alpha_{i,\ell})_{\ell \leq k_i}\right)_{i \leq M-1}, \\ \hat{f}_M \end{array} \right).$$

The verifier samples $r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}}$ and we set $\texttt{State}(\texttt{tr} \| (r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}})) = 1$ if and only if the following all hold:

(a) $\hat{f}_M(\boldsymbol{r}_\ell^{\mathrm{fin}}) = g_{M-1}(r_{M-1}^{\mathrm{fin}})$ for $\ell \in [t_M]$.

(b) $\sum_{\boldsymbol{b}\in\mathbb{B}^{m_M}} \hat{w}_{M-1,k_{M-1}}(\hat{f}_M, \boldsymbol{b}) = \sigma_{M-1,k_{M-1}}$.

(c) $(\mathtt{tr}\|r_1^{\text{fin}}, \ldots, r_{t_M}^{\text{fin}})$ is sumcheck-valid[8].

(We check that folding was done correctly and that all verifier's sumchecks are satisfied.)

5.4.3 Bounding the errors

We bound the probabilities of the state function's flipping from 0 to 1 in every stage of the protocol in accordance with Definition 4.4.

1. **Initial sumcheck at iteration** $s \in [k_0]$. Suppose that in this iteration, we have $\mathtt{State}(\mathtt{tr}) = 0$ (note that if $s = 1$, this means that $f_0 \notin \mathcal{C}$). We show that then

$$\varepsilon_{0,s}^{\text{fold}} = \Pr_{\alpha_{0,s}\leftarrow\$\mathbb{F}}(\mathtt{State}(\mathtt{tr}\|\alpha_{0,\mathtt{s}}) = 1) \leq \frac{d^*\ell_{0,s-1}}{|\mathbb{F}|} + \mathtt{err}^\star(\mathcal{C}_{\text{RS}}^{(0,s)}, 2, \delta_0).$$

If $\mathtt{State}(\mathtt{tr}) = 0$ because $\mathtt{tr}$ is not sumcheck-valid, then $(\mathtt{tr}\|\alpha_{0,s})$ is clearly also not sumcheck-valid for any $\alpha_{0,s}$, which implies that $\mathtt{State}(\mathtt{tr}\|\alpha_{0,s}) = 0$. We therefore assume that $\mathtt{tr}$ is sumcheck-valid, from which follows that

$$\sum_{b\in\mathbb{B}} \hat{h}_{0,s-1}(b) = \sigma_{0,s-1}.$$

This means Item 1b had to fail for $s-1$, i.e. that

$$|\operatorname{List}(\mathcal{C}_{\text{CRS}}^{(0,s-1)}, f_{0,s-1}, \delta_0)| = 0 \tag{7}$$

(and since we assumed that $\delta_0 < \Delta(f_0, \mathcal{C})$, this also holds for $s = 1$). In other words, there is no code word in $\mathcal{C}_{\text{RS}}^{(0,s-1)}$ that is δ_0-close to $f_{0,s-1}$ and satisfies the sumcheck constraint. By Theorem 3.41, since $\delta_0 < 1 - \mathtt{B}^\star(\mathcal{C}_{\text{RS}}^{(0,s)}, 2)$, we have

$$\Pr_{\alpha_{0,s}\leftarrow\$\mathbb{F}}\left[\begin{array}{c}\operatorname{Fold}(\operatorname{List}_b(\mathcal{C}_{\text{RS}}^{(0,s-1)}, 1, f_{0,s}, \delta_0), \alpha_{0,s}) \\ \neq \\ \operatorname{List}(\mathcal{C}_{\text{RS}}^{(0,s)}, \operatorname{Fold}(f_{0,s}, \alpha_{0,s}), \delta_0)\end{array}\right] < \mathtt{err}^\star(\mathcal{C}_{\text{RS}}^{(0,s)}, 2, \delta_0), \tag{8}$$

[8]Recall that sumcheck-validity does not include checks at the final randomness stage.

that is, w.h.p., the list of the code words of $\mathcal{C}_{RS}^{(0,s)}$ that are δ_0-close to $f_{0,s}$ is exactly the folding of the list of the code words of $\mathcal{C}_{\mathrm{RS}}^{(0,s-1)}$ whose block distance is at most δ_0 from $f_{0,s-1}$. It remains for us to show that w.h.p. a code word of $\mathcal{C}_{\mathrm{RS}}^{(0,s-1)}$ that does not satisfy the sumcheck constraint (i.e. is not in $\mathcal{C}_{\mathrm{CRS}}^{(0,s-1)}$) will not fold into a code word of $\mathcal{C}_{\mathrm{RS}}^{(0,s)}$ that does (i.e. into a code word of $\mathcal{C}_{\mathrm{CRS}}^{(0,s)}$). We bound this probability with the next proposition.

Proposition 5.4 ([ACFY25] Claim 5.3)**.**

$$\Pr_{\alpha_{0,s}\leftarrow\$\mathbb{F}}\left[\begin{array}{c}\exists u\in \mathrm{List}_b\left(\mathcal{C}_{\mathrm{RS}}^{(0,s-1)},1,f_{0,s-1},\delta_0\right),\\ \mathrm{Fold}(u,\alpha_{0,s})\in \mathrm{List}\left(\mathcal{C}_{\mathrm{CRS}}^{(0,s)},\mathrm{Fold}(f_{0,s-1},\alpha_{0,s}),\delta_0\right)\end{array}\right]\leq \frac{d^*\ell_{0,s-1}}{|\mathbb{F}|}.$$

Proof. Consider $u\in \mathrm{List}_b\left(\mathcal{C}_{\mathrm{RS}}^{(0,s-1)},1,f_{0,s-1},\delta\right)$ and the degree d^* univariate polynomial

$$\hat{p}(X):=\sum_{\boldsymbol{b}\in\mathbb{B}^{m_0-s}}\hat{w}_{0,s-1}(\hat{u}(X,\boldsymbol{b}),X,\boldsymbol{b}).$$

Since $\left|\mathrm{List}\left(\mathcal{C}_{\mathrm{CRS}}^{(0,s-1)},1,f_{0,s},\delta_0\right)\right|=0$ by (7), it must be that

$$\sum_{b\in\mathbb{B}}\hat{p}(b)=\sum_{\boldsymbol{b}\in\mathbb{B}^{m_0-s+1}}\hat{w}_{0,s-1}(\hat{u}(\boldsymbol{b}),\boldsymbol{b})\neq\sigma_{0,s-1}=\sum_{b\in\mathbb{B}}\hat{h}_{0,s-1}(b).$$

We conclude that $\hat{p}$ and $\hat{h}_{0,s}$ are not identical polynomials: $\hat{p}(X)\not\equiv\hat{h}_{0,s-1}(X)$. The polynomials are both univariate polynomials of degree smaller than d^*, and so by Corollary 2.7,

$$\Pr_{\alpha_{0,s}\leftarrow\$\mathbb{F}}[\hat{p}(\alpha_{0,s})=\hat{h}_{0,s-1}(\alpha_{0,s})]\leq\frac{d^*}{|\mathbb{F}|}.$$

Whenever $\alpha_{0,s}$ is chosen such that this is not the case, we have

$$\sum_{\boldsymbol{b}\in\mathbb{B}^{m_0-s}}\hat{w}_{0,s}(\hat{u}(\alpha_{0,s},\boldsymbol{b}),\boldsymbol{b})=\hat{p}(\alpha_{0,s})\neq\hat{h}_{0,s-1}(\alpha_{0,s})=\sigma_{0,s}$$

(we used $\hat{w}_{0,s}(Z,\boldsymbol{b})=\hat{w}_{0,s-1}(Z,\alpha_{0,s},\boldsymbol{b})$).

Thus, $\mathrm{Fold}(u,\alpha_{0,s})\notin\mathcal{C}_{\mathrm{CRS}}^{(0,s)}$. By Proposition 3.39 and the definition of $\ell_{0,s-1}$,

$$|\,\mathrm{List_b}(\mathcal{C}_{\mathrm{RS}}^{(0,s-1)},1,f_{0,s-1},\delta_0)|\leq|\,\mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(0,s-1)},1,f_{0,s-1},\delta_0)|.$$

Taking the union bound over all the choices of u now yields the desired inequality. □

The bound on the round-by-round soundness error comes by taking a union bound over the probabilities in (8) and Proposition 5.4, which together imply that except with probability $\frac{d^*\ell_{0,s-1}}{|\mathbb{F}|} + \mathtt{err}^\star\left(\mathcal{C}_{\mathrm{RS}}^{(0,s)}, 2, \delta_0\right)$,

$$\mathrm{List}\left(\mathcal{C}_{\mathrm{CRS}}^{(0,s)}, \mathrm{Fold}(f_{0,s-1}, \alpha_{0,s}), \delta_0\right) \cap \mathrm{List}\left(\mathcal{C}_{\mathrm{RS}}^{(0,s)}, \mathrm{Fold}(f_{0,s-1}, \alpha_{0,s}), \delta_0\right) = \emptyset,$$

and so, since

$$\mathrm{List}\left(\mathcal{C}_{\mathrm{CRS}}^{(0,s)}, \mathrm{Fold}(f_{0,s-1}, \alpha_{0,s}), \delta_0\right) \subseteq \mathrm{List}\left(\mathcal{C}_{\mathrm{RS}}^{(0,s)}, \mathrm{Fold}(f_{0,s-1}, \alpha_{0,s}), \delta_0\right),$$

and $f_{0,s} = \mathrm{Fold}\left(f_{0,s-1}, \alpha_{0,s}\right)$ we have

$$\left|\mathrm{List}\left(\mathcal{C}_{\mathrm{CRS}}^{(0,s)}, \mathrm{Fold}(f_{0,s-1}, \alpha_{0,s}), \delta_0\right)\right| = 0,$$

so $\mathtt{State}\left(\mathtt{tr} \| \alpha_{0,s}\right) = 0$.

2. **Out-of-domain sample at iteration** i. We show that if in this stage $\mathtt{State}(\mathtt{tr}) = 0$, then

$$\varepsilon_i^{\mathrm{out}} = \Pr_{z_{i,0} \leftarrow \$ \mathbb{F}}[\mathtt{State}(\mathtt{tr} \| z_{i,0}) = 1] \leq \frac{2^{m_i}\ell_{i,0}^2}{2|\mathbb{F}|}.$$

If $\mathtt{State}(\mathtt{tr}) = 0$, then Item 2a does not hold, as this is exactly what was checked at the previous stage of the state function (and $(\mathtt{tr} \| z_{i,0})$ is sumcheck-valid if and only if $\mathtt{tr}$ is sumcheck-valid). Thus, in order to have $\mathtt{State}(f_0, \mathtt{tr} \| z_{i,0}) = 1$, it must be that Item 2b holds, i.e., there exist distinct $u, u' \in \mathrm{List}(\mathcal{C}_{\mathrm{CRS}}^{(i,0)}, f_i, \delta_i)$ with $\hat{u}\left(\boldsymbol{z}_{i,0}\right) = \hat{u}'\left(\boldsymbol{z}_{i,0}\right)$. By Lemma 3.43, this happens with probability at most $\frac{2^{m_i}\ell_{i,0}^2}{2|F|}$.

3. **Shift message at iteration** i. We show that if $\mathtt{State}\left(\mathtt{tr}\right) = 0$ then

$$\varepsilon_i^{\mathtt{shift}} = \Pr[\mathtt{State}(\mathtt{tr} \| (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i)) = 1] \leq (1 - \delta_{i-1})^{t_i} + \frac{\ell_{i,0}(t_i + 1)}{|\mathbb{F}|}.$$

We assume that $\mathtt{State}\left(f_0, \mathtt{tr}\right) = 0$. This means that both of the following

are true:

(a) there are no distinct $u, u' \in \mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ for which $\hat{u}(\boldsymbol{z}_{i,0}) = \hat{u}'(\boldsymbol{z}_{i,0})$.

(b) at least one of the following holds:

i. $|\mathrm{List}(\mathcal{C}_{\mathrm{CRS}}^{(i-1,k_{i-1})}, g_{i-1}, \delta_{i-1})| = 0$, or

ii. $\mathtt{tr}$ is not sumcheck-valid.

We begin by showing that if all of the code words that are within the list-decoding radius of f_i have some point in which they disagree with the required value, then w.h.p. the state will remain 0:

Proposition 5.5 ([ACFY25] Claim 5.4)**.** *Fix $z_{i,1}, \ldots, z_{i,t_i}$ such that for every $u \in \mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ one of the following hold:*

- $\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{w}_{i-1,k_{i-1}}(\hat{u}(\boldsymbol{b}), \boldsymbol{b}) \neq \sigma_{i-1,k_{i-1}}$, *or*
- $\hat{u}(\boldsymbol{z}_{i,0}) \neq y_{i,0}$, *or*
- *there exists $j \in [t_i]$ for which $\hat{u}(\boldsymbol{z}_{i,j}) \neq g_{i-1}(z_{i,j})$.*

Then $\Pr_{\gamma_i \leftarrow \$\mathbb{F}}[\mathtt{State}(\mathtt{tr} \| (z_{i,1}, \ldots, z_{i,t_i}, \gamma_i)) = 1] \leq \dfrac{\ell_{i,0}(t_i + 1)}{|\mathbb{F}|}$.

Proof. Fix $u \in \mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ and denote $y_{i,j} := g_{i-1}(z_{i,j})$. Then by the claim setup, either $\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{w}_{i-1}(\hat{u}(\boldsymbol{b}), \boldsymbol{\alpha}_{i-1}, \boldsymbol{b}) \neq \sigma_{i-1,k_{i-1}}$ or there exists $j \in \{0, \ldots, t_i\}$ for which

$$\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{u}(\boldsymbol{b}) \, \mathsf{eq}(\boldsymbol{z}_{i,j}, \boldsymbol{b}) = \hat{u}(\boldsymbol{z}_{i,j}) \neq y_{i,j}.$$

This means that the degree $t_i + 1$ (in γ_i) polynomial

$$\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{w}_{i-1}(\hat{u}(\boldsymbol{b}), \boldsymbol{\alpha}_{i-1}, \boldsymbol{b}) + \sum_{j=0}^{t_i} \gamma_i^{j+1} \left(\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{u}(\boldsymbol{b}) \, \mathsf{eq}(\boldsymbol{z}_{i,j}, \boldsymbol{b}) \right)$$

differs from

$$\sigma_{i-1,k_{i-1}} + \sum_{j=0}^{t_i} \gamma_i^{j+1} y_{i,j}.$$

Thus, by the Schwartz–Zippel Lemma 2.7

$$\Pr\left[\sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{w}_i(\hat{u}(\boldsymbol{b}),\boldsymbol{b}) = \sigma_i\right] = \Pr_{\gamma_i\leftarrow\$\mathbb{F}}\left[\begin{array}{c}\sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{w}_i(\hat{u}(\boldsymbol{b}),\boldsymbol{b}) \\ = \\ \sigma_{i-1,k_{i-1}} + \sum_{j=0}^{t_i} \gamma_i^{j+1} y_{i,j}\end{array}\right]$$

$$= \Pr_{\gamma_i\leftarrow\$\mathbb{F}}\left[\begin{array}{c}\sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{w}_{i-1}(\hat{u}(\boldsymbol{b}),\boldsymbol{\alpha}_{i-1},\boldsymbol{b}) \\ + \\ \sum_{j=0}^{t_i} \gamma_i^{j+1}\left(\sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{u}(\boldsymbol{b})\,\mathsf{eq}(\boldsymbol{z}_{i,j},\boldsymbol{b})\right) \\ = \\ \sigma_{i-1,k_{i-1}} + \sum_{j=0}^{t_i} \gamma_i^{j+1} y_{i,j}\end{array}\right]$$

$$\leq \frac{t_i+1}{|\mathbb{F}|}.$$

Taking the union bound over the at most $\ell_{i,0}$ code words in $\mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_{i,0}, \delta_i)$, we have no $u \in \mathrm{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ for which

$$\sum_{\boldsymbol{b}\in\mathbb{B}^{m_i}} \hat{w}_i(\hat{u}(\boldsymbol{b}),\boldsymbol{b}) = \sigma_{i-1,k_{i-1}} + \sum_{j=0}^{t_i} \gamma_i^{j+1} y_{i,j} = \sigma_{i,0},$$

except with probability $\dfrac{\ell_{i,0}(t_i+1)}{|\mathbb{F}|}$.

Recall that $\mathcal{C}_{\mathrm{CRS}}^{(i,0)} = \mathrm{CRS}[\mathbb{F}, \mathcal{L}_i, m_i, \hat{w}_i, \sigma_{i,0}]$ and so we conclude that, except with probability $\dfrac{\ell_{i,0}(t_i+1)}{|\mathbb{F}|}$ it is the case that $|\,\mathrm{List}(\mathcal{C}_{\mathrm{CRS}}^{(i,0)}, f_i, \delta_i)| = 0$, and so the state is 0. □

We now show that the requirements for Proposition 5.5 hold except with probability $(1-\delta_{i-1})^{t_i}$ over the choice of $z_{i,1}, \ldots, z_{i,t_i}$. Item 3b in the state function definition posits that in order to have

$$\mathtt{State}(\mathtt{tr}\|(z_{i,1},\ldots,z_{i,t_i},\gamma_i)) = 1,$$

it must be that $\mathtt{tr}\|\,(z_{i,1},\ldots,z_{i,t_i},\gamma_i)$ is sumcheck-valid, meaning $\mathtt{tr}$ has to

also be sumcheck-valid. Thus, the only chance for the state to switch to 1 is in the case that $\mathtt{tr}$ is sumcheck-valid. The only way to have $\mathtt{State}(\mathtt{tr}) = 0$ when $\mathtt{tr}$ is sumcheck-valid is when $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i-1,k_{i-1})}, g_{i-1}, \delta_{i-1})| = 0$. There are three cases:

- There is no $u \in \operatorname{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ for which $\hat{u}(\boldsymbol{z}_{i,0}) = y_{i,0}$. In this case the requirements for Proposition 5.5 hold.
- There is exactly one $u \in \operatorname{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ for which $\hat{u}(\boldsymbol{z}_{i,0}) = y_{i,0}$. Recall that $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(i-1,k_{i-1})}, g_{i-1}, \delta_{i-1})| = 0$. Thus, u either disagrees with the constraint defined by $\hat{w}_{i-1,k_{i-1}}$ or u is δ_{i-1}-far from g_{i-1}:
  - if $\sum_{\boldsymbol{b} \in \mathbb{B}^{m_i}} \hat{w}_{i-1,k_{i-1}}(\hat{u}(\boldsymbol{b}), \boldsymbol{b}) \neq \sigma_{i-1,k_{i-1}}$, then the requirements for Proposition 5.5 hold.
  - if $\Delta(\hat{u}|_{\mathcal{L}_{i-1}}, g_{i-1}) > \delta_{i-1}$ then the probability that for all points $z_{i,j}$, $j \in [t_i]$, we have $\hat{u}(\boldsymbol{z}_{i,j}) = g_{i-1}(z_{i,j})$ is at most $(1-\delta_{i-1})^{t_i}$. Since $\hat{v}(\boldsymbol{z}_{i,0}) \neq y_{i,0}$ for every $v \in \operatorname{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ with $v \neq u$, no other code word suits. We conclude that except with probability $(1-\delta_{i-1})^{t_i}$, the requirements for Proposition 5.5 hold.
- There are multiple code words $u \in \operatorname{List}(\mathcal{C}_{\mathrm{RS}}^{(i,0)}, f_i, \delta_i)$ for which $\hat{u}(\boldsymbol{z}_{i,0}) = y_{i,0}$. We ruled this out by the fact that $\mathtt{State}(\mathtt{tr}) = 0$ in Item 3a.

4. **Folding randomness at round s of iteration i.** We show that if $\mathtt{State}(\mathtt{tr}) = 0$ then

$$\varepsilon_{i,s}^{\mathrm{fold}} = \Pr_{\alpha_{i,s} \leftarrow \$\mathbb{F}}[\mathtt{State}\left(\mathtt{tr} \| \boldsymbol{\alpha}_i\right) = 1] \leq \frac{d\ell_{i,s-1}}{|\mathbb{F}|} + \mathtt{err}^{\star}\left(\mathcal{C}_{\mathrm{RS}}^{(i,s)}, 2, \delta_i\right).$$

   The analysis is identical to that in the case of the initial sumcheck with index i in place of 0, and degree d in place of d^*.

5. **Final randomness.** We show that if $\mathtt{State}(\mathtt{tr}) = 0$ then

$$\varepsilon^{\mathrm{fin}} = \Pr_{r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}} \leftarrow \$\mathbb{F}}[\mathtt{State}(\mathtt{tr} \| (r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}})) = 1] \leq (1-\delta_{M-1})^{t_M}.$$

   If $\mathtt{tr}$ is not sumcheck-valid then then $(\mathtt{tr} \| (r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}}))$ is also not sumcheck-valid and so $\mathtt{State}(\mathtt{tr} \| (r_1^{\mathrm{fin}}, \ldots, r_{t_M}^{\mathrm{fin}})) = 0$. Thus, we can assume

that `tr` is sumcheck-valid. Then, it must be that Item 4a does not hold, and as such $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(M-1,k_{M-1})}, g_{M-1}, \delta_{M-1})| = 0$. We consider two options.

- $\Delta(\hat{f}_M, g_{M-1}) \leq \delta_{M-1}$. Let f_M be the restriction of $\hat{f}_M$ to the domain of $\mathcal{C}_{\mathrm{CRS}}^{(M-1,k_{M-1})}$. Then $f_M \in \operatorname{List}(\mathcal{C}_{\mathrm{RS}}^{(M-1,k_{M-1})}, g_{M-1}, \delta_{M-1})$. Since $|\operatorname{List}(\mathcal{C}_{\mathrm{CRS}}^{(M-1,k_{M-1})}, g_{M-1}, \delta_{M-1})| = 0$, it must be that the sumcheck condition fails, i.e.,

$$\sum_{\boldsymbol{b} \in \mathbb{B}^{m_M}} \hat{w}_{M-1,k_{M-1}}(\hat{f}_M(\boldsymbol{b}), \boldsymbol{b}) \neq \sigma_{M-1,k_{M-1}}.$$

  But this contradicts Item 5b, and so

$$\texttt{State}(\texttt{tr} \| (r_1^{\mathrm{fin}}, \ldots, r_{t_{M-1}}^{\mathrm{fin}})) = 0.$$

- $\Delta(\hat{f}_M, g_{M-1}) > \delta_{M-1}$. Then $\hat{f}_M$ agrees with g_{M-1} at t_M randomly selected locations with probability at most $(1 - \delta_{M-1})^{t_M}$, and if this does not occur then Item 5a does not hold.

6. **Verifier decision**. If $\texttt{State}(\texttt{tr}) = 0$, then the verifier rejects, since the verifier's checks align precisely with the definition of the state function at the final randomness stage.

This concludes the proof of Theorem 5.3.

5.4.4 Choosing parameters

It was shown in [BCS16] that in order for the final hash-based zk-SNARK to have λ-bit security, we need all the rounds of the IOPP to have λ-bit security.

We analyze the parameters of Construction 5.1 in the unique decoding regime. Let $n := |\mathcal{L}_0|$ and $|\mathcal{L}_i| = \dfrac{n}{2^i}$. Let $\rho_{i,s}$ be the code rate of $\mathcal{C}_{\mathrm{RS}}^{(i,s)}$. Then

$$\rho_i := \rho_{i,s} = \frac{2^{m_i - s}}{2^{-i-s} n} = \frac{2^{m_i + i}}{n}.$$

By Corollary 3.32, $\texttt{Gen}(\ell; \alpha) = (1, \alpha, \ldots, \alpha^{\ell-1})$ is a proximity generator with mutual

correlated agreement for each of the codes $(\mathcal{C}_{\mathrm{RS}}^{(i,s)})_{\substack{0\le i\le M-1\\ 0\le s\le k_i}}$ with proximity bound

$$\mathtt{B}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,s)},\ell)=\frac{1+\rho_i}{2}=\frac{1}{2}+\frac{2^{m_i+i-1}}{n}$$

and error

$$\mathtt{err}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,s)},\ell,\delta)=\frac{2^{m_i-s}(\ell-1)}{\rho_i|\mathbb{F}|}=\frac{(\ell-1)n}{2^{i+s}|\mathbb{F}|}.$$

We choose δ_i to be the unique decoding distance for each $i\in\{0,\dots,M-1\}$:

$$\delta_i=\frac{1-\rho_i}{2}=1-\mathtt{B}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,0)},2)=\frac{1}{2}-\frac{2^{m_i+i-1}}{n}$$

(formally, we need to take $\delta_i=1-\mathtt{B}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,0)},2))-\varepsilon$ for a negligibly small ε). In the unique decoding regime, we have $\ell_{i,s}=1$ for all $i\in\{0,1,\dots,M-1\}$ and $s\in\{0,1,\dots,k_i\}$.

We now calculate the errors in each round.

- $\varepsilon_{0,s}^{\mathrm{fold}}\le\dfrac{d^*\ell_{0,s-1}}{|\mathbb{F}|}+\mathtt{err}^\star(\mathcal{C}_{\mathrm{RS}}^{(0,s)},2,\delta_0)=\dfrac{d^*+2^{-s}n}{|\mathbb{F}|}$;
- $\varepsilon_i^{\mathrm{out}}\le\dfrac{2^{m_i}\ell_{i,0}^2}{2|\mathbb{F}|}=\dfrac{2^{m_i-1}}{|\mathbb{F}|}$;
- $\varepsilon_i^{\mathrm{shift}}\le(1-\delta_{i-1})^{t_i}+\dfrac{\ell_{i,0}(t_i+1)}{|\mathbb{F}|}=\left(\dfrac{1}{2}+\dfrac{2^{m_{i-1}+i-2}}{n}\right)^{t_i}+\dfrac{t_i+1}{|\mathbb{F}|}$;
- $\varepsilon_{i,s}^{\mathrm{fold}}\le\dfrac{d\ell_{i,s-1}}{|\mathbb{F}|}+\mathtt{err}^\star(\mathcal{C}_{\mathrm{RS}}^{(i,s)},2,\delta_i)=\dfrac{d+2^{-i-s}n}{|\mathbb{F}|}$;
- $\varepsilon^{\mathrm{fin}}\le(1-\delta_{M-1})^{t_M}=\left(\dfrac{1}{2}+\dfrac{2^{m_{M-1}+M-2}}{n}\right)^{t_M}$.

In usual applications, $|\mathbb{F}|\gg m_0$ (e.g. $|\mathbb{F}|\approx 2^{192}$ and $m_0=2^{24}$). We take arbitrary smooth evaluation domains $\mathcal{L}_0,\dots,\mathcal{L}_{M-1}$ with $|\mathcal{L}_0|=n=\dfrac{2^{m_0}}{\rho_0}$. Usually ρ_0 is between 2^{-5} and 2^{-1}.

Because of this, the folding and out-of-domain sample errors are negligible compared to the shifting errors and we consider only these.

To have $\varepsilon_i^{\mathrm{shift}}<2^{-\lambda}$ and $\varepsilon^{\mathrm{fin}}<2^{-\lambda}$, we must have

$$t_i>\frac{\lambda}{-\log_2(1-\delta_{i-1})}=\frac{\lambda}{1-\log_2(1+\rho_{i-1})}$$

for $i \in \{1, \ldots, M-1\}$. The Maclaurin series expansion gives the approximation $\log_2(1+\rho_{i-1}) \approx \frac{\rho_{i-1}}{\ln 2}$ for $\rho_{i-1} \ll 1$. This allows us to write

$$t_i > \frac{\lambda}{1 - \log_2(1+\rho_{i-1})} \approx \frac{\lambda}{\ln 2 - \ln \rho_{i-1}} = \frac{\lambda}{\ln(2/\rho_{i-1})}.$$

Asymptotically,

$$t_i = O\left(\frac{\lambda}{\log_2(1/\rho_{i-1})}\right).$$

Finally, we note that by looking at the error bounds in Theorem 5.3, we see that larger values of δ_i (and smaller values of ρ_i) allow for smaller values of t_i (given that the $\ell_{i,s}$ stay comparatively small which holds at least to the Johnson bound). But to choose δ_i larger that $\frac{1-\rho_i}{2}$, we need to assume that Conjecture 3.33 holds.

5.5 Batching multiple constraints

We saw in Chapter 5.1, that constraints on the Reed-Solomon code correspond to a verifier challenges. It is therefore very often that we use multi-constrained RS codes: instead of running an instance of the costly IOPP for each verifier challenge, we batch the challenges together and run WHIR once. In this chapter, we show how to do this and prove its round-by-round soundness.

Construction 5.6 ([ACFY25] Construction 5.5**).**

- Ingredients and notation.
  - $(\mathcal{P}_1, \mathcal{V}_1)$ is a k-round IOPP for smooth CRS codes $\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \cdot, \cdot]$ (e.g. WHIR);
  - $t \in \mathbb{N}$ is the number of weights (constraints);
  - $\hat{w}_1, \ldots \hat{w}_t \in \mathbb{F}[Z, X_1, \ldots, X_m]$ are weights;
  - $\sigma_1, \ldots, \sigma_t \in \mathbb{F}$ are targets.
- **Goal**. The goal of the protocol is for the prover $\mathcal{P}$ to convince the verifier $\mathcal{V}$ that $f\colon \mathcal{L} \to \mathbb{F}$ is δ-close to a code word of the multi-constrained code $\mathcal{C} = \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_1, \sigma_1), \ldots, (\hat{w}_t, \sigma_t)]$.
- **Inputs**. In the honest case, $\mathcal{V}$ has oracle access to f and $\mathcal{P}$ receives $\hat{f}$ such that $\hat{f}|_{\mathcal{L}} = f$ and $f \in \mathcal{C}$.

- **Interaction phase.**
  1. $\mathcal{V}$ samples $\gamma \leftarrow\!\!\$\ \mathbb{F}$ and sends it to $\mathcal{P}$.
  2. We define
  $$\hat{w} := \sum_{i\in[t]} \gamma^{i-1}\hat{w}_i(Z, X_1, \ldots, X_m)$$
  and $\sigma := \sum_{i\in[t]} \gamma^{i-1}\sigma_i$.

  The $\mathcal{P}$ and $\mathcal{V}$ engage in the interaction phase of $(\mathcal{P}_1, \mathcal{V}_1)$ to check that $f \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{w}, \sigma]$.

- **Decision phase**. $\mathcal{V}$ engages in the decision stage of $(\mathcal{P}_1, \mathcal{V}_1)$ with access to f.

The parameters are the same as of the chosen IOPP $(\mathcal{P}_1, \mathcal{V}_1)$, except verifier sends one additional randomly sampled element $\gamma \in \mathbb{F}$.

5.5.1 Round-by-round soundness

We prove that Construction 5.6 has round-by-roundness.

Theorem 5.7 ([ACFY25] Theorem 5.6)**.**
Let $(\mathbb{F}, \mathcal{L}, m, \mathcal{P}_1, \mathcal{V}_1, k, t, (\hat{w}_1, \sigma_1), \ldots, (\hat{w}_t, \sigma_t))$ be as in Construction 5.6. Suppose that the IOPP $(\mathcal{P}_1, \mathcal{V}_1)$ has round-by-round soundness with error $(\mathtt{err}_1^1, \ldots, \mathtt{err}_1^k)$ and that $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ is (δ, ℓ)-list decodable.

Then Construction 5.6 is an IOPP for the multi-constrained Reed-Solomon code $\mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \mathcal{P}_1, \mathcal{V}_1, k, t, (\hat{w}_1, \sigma_1), \ldots, (\hat{w}_t, \sigma_t)]$ with round-by-roundness error

$$\left(\frac{t-1}{|\mathbb{F}|}, \mathtt{err}_1^1(\delta), \ldots, \mathtt{err}_1^k(\delta)\right)$$

for proximity parameter $\delta \in [0, 1]$.

Proof. Let $\mathtt{State}_1$ be the state function of $(\mathcal{P}_1, \mathcal{V}_1)$. We first define a state function and then prove that it has the required round-by-round soundness error.

0. **Initial transcript**. We set $\mathtt{State}(f, (\hat{w}_i, \sigma_i)_{i\in[t]}, \emptyset) = 1$ if and only if
$$f \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_i, \sigma_i)_{i\in[t]}].$$

1. **Combination interaction**. The verifier samples γ. We set

$$\texttt{State}(f, (\hat{w}_i, \sigma_i)_{i\in[t]}, \gamma) = 1$$

if and only if $f \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{w}, \sigma]$.

2. **Proximity test rounds**. In round i of the proximity test, let

$$\texttt{tr} = (\gamma, \alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$$

with the α_j, $j \in [i]$, being prover's messages and the $\beta_j, j \in [i-1]$, verifier's challenges.

The verifier chooses β_i. We set

$$\texttt{State}(f, (\hat{w}_i, \sigma_i)_{i\in[t]}, \texttt{tr}\|\beta_i) = \texttt{State}_1(f, \hat{w}, \sigma, \alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i, \beta_i).$$

We now analyze the round-by-round soundness error with respect to the above state function. The proof of [ACFY25, Theorem 5.6] does not correctly justify the soundness of the combination interaction. We therefore give a new proof of this theorem. The revised argument leads to a soundness error that is strictly smaller than the bound stated in [ACFY25].

1. **Combination interaction**. If $\texttt{State}(f, (\hat{w}_i, \sigma_i)_{i\in[t]}, \emptyset) = 0$ then $f \notin \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_j, \sigma_j)_{j\in[t]}]$. If $f \notin \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ then clearly $f \notin \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, \hat{w}, \sigma]$, and so $\texttt{State}(f, (\hat{w}_i, \sigma_i)_{i\in[t]}, \gamma) = 0$ regardless of γ.

Suppose then that $f \in \mathrm{RS}[\mathbb{F}, \mathcal{L}, m]$ and thus $f = \hat{f}|_{\mathcal{L}}$ for a low-degree polynomial $\hat{f}$. Since $f \notin \mathrm{CRS}[\mathbb{F}, \mathcal{L}, m, (\hat{w}_i, \sigma_i)_{i\in[t]}]$, we must have an $i \in [t]$ such that

$$\sum_{\boldsymbol{b}\in\mathbb{B}^m} \hat{w}_i(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) \neq \sigma_i.$$

This means that the degree $t-1$ polynomials (in γ)

$$\sum_{\boldsymbol{b}\in\mathbb{B}^m} \hat{w}(\hat{f}(\boldsymbol{b}), \boldsymbol{b}) = \sum_{i\in[t]} \gamma^{i-1} \sum_{\boldsymbol{b}\in\mathbb{B}^m} \hat{w}_i(\hat{f}(\boldsymbol{b}), \boldsymbol{b})$$

and

$$\sigma = \sum_{i\in[t]} \gamma^{i-1}\sigma_i$$

are different. Thus, by Corollary 2.7:

$$\Pr_{\gamma\leftarrow\$\mathbb{F}}\left[\sum_{\boldsymbol{b}\in\mathbb{B}^m}\hat{w}(\hat{f}(\boldsymbol{b}),\boldsymbol{b})=\sigma\right]=\Pr_{\gamma\leftarrow\$\mathbb{F}}\left[\begin{array}{c}\sum_{i\in[t]}\gamma^{i-1}\sum_{\boldsymbol{b}\in\mathbb{B}^m}\hat{w}_i(\hat{f}(\boldsymbol{b}),\boldsymbol{b})\\ =\\ \sum_{i\in[t]}\gamma^{i-1}\sigma_i\end{array}\right]\leq\frac{t-1}{|\mathbb{F}|}.$$

2. **Proximity test rounds**. Let $i=1$. Then $\mathtt{tr}=(\gamma,\alpha_1)$. We assume that $\mathtt{State}(f,(\hat{w}_i,\sigma_i)_{i\in[t]},\mathtt{tr})=0$, implying that $f\notin\mathrm{CRS}[\mathbb{F},\mathcal{L},m,\hat{w},\sigma]$ so $\mathtt{State}_1(f,\hat{\omega},\sigma,\emptyset)=0$. Then

$$\begin{aligned}\Pr_{\beta_1\leftarrow\mathcal{V}_1}\left(\mathtt{State}(f,(\hat{\omega}_i,\sigma_i)_{i\in[t]},\mathtt{tr}\|\beta_1)=1\right)&=\Pr_{\beta_1\leftarrow\$\mathbb{F}}\left(\mathtt{State}_1(f,\hat{\omega},\sigma,\alpha_1,\beta_1)=1\right)\\&=\mathtt{err}_1^1(\delta).\end{aligned}$$

Now let $i\geq 2$. Then

$$\mathtt{State}(f,(\hat{w}_i,\sigma_i)_{i\in[t]},\gamma,(\alpha_j,\beta_j)_{j\in[i-1]})=0$$

implies

$$\mathtt{State}_1(f,\hat{\omega},\sigma,\gamma,(\alpha_j,\beta_j)_{j\in[i-1]})=0,$$

and so

$$\begin{aligned}&\Pr_{\beta_i\leftarrow\mathcal{V}_1}[\mathtt{State}(f,(\hat{w}_i,\sigma_i)_{i\in[t]},\gamma,(\alpha_j,\beta_j)_{j\in[i]})=1]\\&\quad=\Pr_{\beta_i\leftarrow\mathcal{V}_1}[\mathtt{State}_1(f,\hat{w},\sigma,(\alpha_j,\beta_j)_{j\in[i]})=1]\leq\mathtt{err}_1^i(\delta).\end{aligned}$$

3. **Verifier decision**. The verifier rejects if $\mathcal{V}_1$ rejects. If

$$\mathtt{State}(f,(\hat{w}_i,\sigma_i)_{i\in[t]},\gamma,\alpha_1,\beta_1,\ldots,\alpha_k,\beta_k)=0$$

then $\mathtt{State}_1(f,\hat{w},\sigma,\alpha_1,\beta_1,\ldots,\alpha_k,\beta_k)=0$ and so $\mathcal{V}_1$ rejects.

□

5.6 Example application of WHIR

We conclude with a concrete example illustrating how WHIR transforms a polynomial IOP (PIOP) into an IOP. For simplicity, we choose very small parameters; in practice, the parameters are selected as functions of the proximity parameter δ and the security parameter λ.

Example 5.8.

Setup.

- *Let* $\mathbb{F} = \mathbb{F}_{17}$. *Let* $m = 3$ *and* $\mathcal{P}$ *have the polynomial*

$$\mathbb{F}^{<2}[X_1, X_2, X_3] \ni \hat{f}_0 := 4X_1X_2 + 2X_2X_3 + 3X_1X_3 - X_1 - 2X_2 - 3X_3.$$

  $\mathcal{V}$ *wishes to verify that* $\hat{f}(0,0,0) + \hat{f}(1,1,1) = 3$.

- *In the IOP,* $\mathcal{V}$ *has oracle access to* $f_0 \colon \mathcal{L}_0 \to \mathbb{F}$ *where* $f_0(x) := \hat{f}_0(\mathrm{pow}(x,3)) = \hat{f}_0(x, x^2, x^4)$ *for* $x \in \mathcal{L}_0$. *In our case,*

$$f_0(x) := 2x^6 + 3x^5 - 3x^4 + 4x^3 - 2x^2 - x.$$

- *We define the weight polynomial and target:*

$$\begin{aligned}\hat{w}_0(Z, X_1, X_2, X_3) :=& Z\,\mathsf{eq}((0,0,0),(X_1,X_2,X_3)) \\ &+ Z\,\mathsf{eq}((1,1,1),(X_1,X_2,X_3)) \\ =& Z(1-X_1)(1-X_2)(1-X_3) + ZX_1X_2X_3 \\ =& Z(X_1X_2 + X_1X_3 + X_2X_3 - X_1 - X_2 - X_3 + 1)\end{aligned}$$

  and $\sigma_0 := 3$.

  Now, since $\hat{f}_0$ *is multilinear,* $\sum_{\boldsymbol{b} \in \mathbb{B}^3} \hat{w}_0(\hat{f}_0(\boldsymbol{b}), \boldsymbol{b}) = \hat{f}_0(0,0,0) + \hat{f}_0(1,1,1)$ *and thus* $f_0 \in \mathrm{CRS}[\mathbb{F}, \mathcal{L}, 3, \hat{w}_0, 3] =: \mathcal{C}$ *if and only if* $\hat{f}_0(0,0,0) + \hat{f}_0(1,1,1) = 3$.

$\mathcal{P}$ *and* $\mathcal{V}$ *now engage in the WHIR protocol to prove that* f_0 *is* δ*-close to a code word of* $\mathcal{C}$. *We choose:*

$$m_0 = 3, \quad M := 2, \quad k_0 = k_1 = 1, \quad t_1 = t_2 = 2.$$

Then

$$m_1 = 2, \quad m_2 = 1.$$

We let

$$\mathcal{L}_0 = \mathcal{L}_1 = \{3, 5, 6, 7, -7, -6, -5, -3\},$$

then

$$\mathcal{L}_0^2 = \mathcal{L}_1^2 = \{2, 8, -8, -2\}.$$

1. ***Initial sumcheck***. *$\mathcal{P}$ sends $\mathcal{V}$ the univariate polynomial*

$$\hat{h}_{0,1}(X) := \sum_{b_1, b_2 \in \mathbb{B}} \hat{w}_0(\hat{f}_0(X, b_1, b_2), X, b_1, b_2) = 7X^2 - 4X.$$

   The verifier samples $\alpha_{0,1} := -2$.

   Since $k_0 = 1$, this completes the initial sumcheck.

2. ***Main loop***.

   (a) *$\mathcal{P}$ calculates*

$$\hat{f}_1(X_1, X_2) := \hat{f}_0(-2, X_1, X_2) = -10X_1 + 2X_1X_2 - 9X_2 + 2$$

   and

$$f_1(x) := \hat{f}_1(x, x^2) = 2x^3 - 9x^2 - 10x + 2.$$

   $\mathcal{P}$ sends $\mathcal{V}$ the oracle to f_1 over $\mathcal{L}_1$.

   (b) *$\mathcal{V}$ samples $z_{1,0} = -3$ and sends it to $\mathcal{P}$. They set*

$$\boldsymbol{z}_{1,0} = \mathrm{pow}(-3, 2) = (-3, 9).$$

   (c) *$\mathcal{P}$ calculates*

$$y_{1,0} := \hat{f}_1(-3, 9) = -1$$

   and sends it to $\mathcal{V}$.

   (d) *$\mathcal{V}$ samples $z_{1,1} = -2$ and $z_{1,2} = 2$ and $\gamma_1 = 5$. They set*

$$\boldsymbol{z}_{1,1} = \mathrm{pow}(-2, 2) = (-2, 4),$$
$$\boldsymbol{z}_{1,2} = \mathrm{pow}(2, 2) = (2, 4).$$

(e) We set

$$\begin{aligned}\hat{w}_1(Z, X_1, X_2) &:= \hat{w}_0(Z, -2, X_1, X_2) + \sum_{j=0}^{2} \gamma^{j+1} Z\, \mathtt{eq}(\boldsymbol{z}_{1,j}, (X_1, X_2)) \\ &= 3ZX_1 + 4ZX_2 + 10Z.\end{aligned}$$

$\mathcal{P}$ sends $\mathcal{V}$ the univariate polynomial

$$\hat{h}_{1,1}(X) := \sum_{b \in \mathbb{B}} \hat{w}_1(\hat{f}_1(X, b), X, b) = -3X^2 - 6X + 7.$$

$\mathcal{V}$ samples $\alpha_{1,1} = 4$.

Since $M - 1 = k_1 = 1$, this concludes the main loop.

3. ***Send final polynomial**. $\mathcal{P}$ sends $\mathcal{V}$ the final polynomial*

$$\hat{f}_2(X) := \hat{f}_1(4, X) = -4 - X.$$

4. ***Sample final randomness**. $\mathcal{V}$ samples final randomness $r_1^{\text{fin}} = -2$ and $r_2^{\text{fin}} = -8$.*

***Decision stage**.*

1. ***Check initial sumcheck**. $\mathcal{V}$ checks that $\hat{h}_{0,1}(0) + \hat{h}_{0,1}(1) = \sigma_0$. This holds since*

$$\hat{h}_{0,1}(0) + \hat{h}_{0,1}(1) = 0 + 3 = 3 = \sigma_0.$$

2. ***Check main loop**. $\mathcal{V}$ has to calculate foldings. She first queries the oracle for evaluations of f_0 at $\text{Block}(\mathcal{L}_0, 1, -2) = \{-7, 7\}$ and $\text{Block}(\mathcal{L}_0, 1, 2) = \{-6, 6\}$:*

x	-7	7	-6	6
$f_0(x)$	6	-3	5	-5

$\mathcal{V}$ *now computes*

$$\begin{aligned} g_0(z_{1,1}) &= \mathrm{Fold}(f_0, -2)(-2) \\ &= \frac{f_0(-7) + f_0(7)}{2} - 2 \cdot \frac{f_0(-7) - f_0(7)}{2 \cdot (-7)} \\ &= 4; \\ g_0(z_{1,2}) &= \mathrm{Fold}(f_0, -2)(2) \\ &= \frac{f_0(-6) + f_0(6)}{2} - 2 \cdot \frac{f_0(-6) - f_0(6)}{2 \cdot (-6)} \\ &= -4. \end{aligned}$$

Using this, $\mathcal{V}$ *checks that*

$$\hat{h}_{1,1}(0) + \hat{h}_{1,1}(1) = \hat{h}_{0,1}(\alpha_{0,1}) + \gamma_1 y_{1,0} + \sum_{j=1}^{t_i} \gamma_1^{j+1} g_0(z_{1,j}).$$

Indeed, $\hat{h}_{1,1}(0) + \hat{h}_{1,1}(1) = 7 - 2 = 5$ *and*

$$\hat{h}_{0,1}(\alpha_{0,1}) + \gamma_1 y_{1,0} + \sum_{j=1}^{2} \gamma_1^{j+1} g_0(z_{1,j}) = 2 + 5(-1) + 8 \cdot 4 + 6(-4) = 5.$$

Since $M - 1 = k_1 = 1$*, this concludes checking the main loop.*

3. ***Check final polynomial.*** $\mathcal{V}$ *queries for evaluations of* f_1 *at* $\mathrm{Block}(\mathcal{L}_0, 1, -2) = \{-7, 7\}$ *and* $\mathrm{Block}(\mathcal{L}_0, 1, -8) = \{-3, 3\}$*:*

x	-7	7	-3	3
$f_1(x)$	-1	7	-1	-4

$\mathcal{V}$ *now computes*

$$\begin{aligned}
g_1(r_1^{\text{fin}}) &= \text{Fold}(f_1, 4)(-2) \\
&= \frac{f_1(-7) + f_1(7)}{2} + 4 \cdot \frac{f_1(-7) - f_1(7)}{2 \cdot (-7)} \\
&= -2; \\
g_1(r_2^{\text{fin}}) &= \text{Fold}(f_1, 4)(-8) \\
&= \frac{f_1(-3) + f_1(3)}{2} + 4 \cdot \frac{f_1(-3) - f_1(3)}{2 \cdot (-3)} \\
&= 4.
\end{aligned}$$

$\mathcal{V}$ *checks that* $\hat{f}_2(r_1^{\text{fin}}) = g_1(r_1^{\text{fin}})$ *and* $\hat{f}_2(r_2^{\text{fin}}) = g_1(r_2^{\text{fin}})$*. And indeed,* $\hat{f}_2(r_1^{\text{fin}}) = -2$ *and* $\hat{f}_2(r_2^{\text{fin}}) = 4$*.*

Finally, $\mathcal{V}$ *verifies that*

$$\sum_{b \in \mathbb{B}} \hat{w}_1(\hat{f}_2(b), 4, b) = 3 = \hat{h}_{1,1}(4).$$

This concludes the execution of the WHIR protocol. Since all verifier checks succeed, the verifier accepts the transaction and that f_0 *is* δ*-close to the code* $\mathcal{C}$*, which in turn implies that the underlying multilinear polynomial* $\hat{f}_0$ *satisfies the constraint* $\hat{f}_0(0,0,0) + \hat{f}_0(1,1,1) = 3$*.*

6 Conclusion

In this thesis, we studied WHIR, an interactive oracle proof of proximity for multilinear codes introduced in [ACFY25]. A central contribution of this work is a detailed analysis of the construction and security of WHIR, together with a corrected and clarified presentation of several arguments that were incomplete or incorrect in the original exposition. We hope that this thesis provides a clearer and more accessible account of the protocol and its underlying techniques.

In Chapter 3, we introduced smooth constrained Reed–Solomon codes, mutual correlated agreement, and block distances. These notions form the algebraic foundation of WHIR and are used throughout the security analysis in Chapter 5. We also proved the Johnson bound in Theorem 3.10, illustrating the list sizes that arise in the list decoding of Reed–Solomon codes.

In Chapter 4, we reviewed the framework of interactive oracle proofs (IOPs), IOPs of proximity (IOPPs), and polynomial IOPs (PIOPs), together with cryptographic tools such as Merkle trees and the Fiat–Shamir transform. These components enable the compilation of a PIOP into a post-quantum secure hash-based zk-SNARK via proximity testing and the BCS transformation.

Finally, in Chapter 5, we presented the construction of WHIR (Construction 5.1) and analyzed its design in detail. We proved its completeness and established its round-by-round soundness, providing corrected proofs for several technical claims and correcting two error bounds in comparison with those stated in [ACFY25]. We finished with Example 5.8, which explicitly demonstrated how WHIR can be applied to transform a concrete PIOP into an IOP.

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight sublinear arguments without a trusted setup". *Proceedings of the 2017 acm sigsac conference on computer and communications security*. 2017, pp. 2087–2104.

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "STIR: Reed–Solomon Proximity Testing with Fewer Queries". *Proceedings of the 44th Annual International Cryptology Conference (CRYPTO '24)*. Lecture Notes in Computer Science. Springer, 2024, pp. 380–413. DOI: 10.1007/978-3-031-36832-3_13.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "WHIR: Reed–Solomon proximity testing with super-fast verification". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2025, pp. 214–243.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP '18)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14.

[BCIKS20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. *Proximity Gaps for Reed-Solomon Codes*. Cryptology ePrint Archive, Paper 2020/654. 2020. URL: https://eprint.iacr.org/2020/654.

[BCKL22] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. "Elliptic Curve Fast Fourier Transform (ECFFT) Part II: Scalable and Transparent Proofs over All Large Fields" (2022).

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive oracle proofs". *Theory of Cryptography Conference*. Springer. 2016, pp. 31–60.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. "Batching techniques for accumulators with applications to IOPs and stateless blockchains". *Annual International Cryptology Conference*. Springer. 2019, pp. 561–586.

[BLNR20] Sarah Bordage, Mathieu Lhotel, Jade Nardi, and Hugues Randriam. "Interactive oracle proofs of proximity to algebraic geometry codes". *arXiv preprint arXiv:2011.04295* (2020).

[BCFRRZ25] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D Rothblum, and Hadas Zeilberger. "Blaze: Fast SNARKs from interleaved RAA codes". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2025, pp. 123–152.

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short proofs for confidential transactions and more". *2018 IEEE symposium on security and privacy (SP)*. IEEE. 2018, pp. 315–334.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK compilers". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 677–706.

[But+14] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". *white paper* 3.37 (2014), pp. 2–1.

[CCHLRRW19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N Rothblum, Ron D Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 1082–1090.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with universal and updatable SRS". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 738–768.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A secure and optimally efficient multi-authority election scheme". *European transactions on Telecommunications* 8.5 (1997), pp. 481–490.

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194.

[GLSTW21] Alexander Golovnev, Jonathan Lee, Srinath TV Setty, Justin Thaler, and Riad S Wahby. "Brakedown: Linear-time and post-quantum SNARKs for R1CS." *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1043.

[GRS25] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. Draft version, August 26, 2025. Available at `http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`. University at Buffalo, SUNY, 2025.

[HLP24] Ulrich Haböck, David Levit, and Shahar Papini. "Circle starks". *Cryptology ePrint Archive* (2024).

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". *Advances in Cryptology – ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5–9, 2010, Proceedings, Part I*. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_10.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868.

[MZ17] Payman Mohassel and Yupeng Zhang. "SecureML: A System for Scalable Privacy-Preserving Machine Learning". *IEEE S&P*. 2017.

[SCGGMTV14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin". *2014 IEEE symposium on security and privacy*. IEEE. 2014, pp. 459–474.

[Sho94] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Tha13] Justin Thaler. "Time-optimal interactive proofs for circuit evaluation". *Annual cryptology conference*. Springer. 2013, pp. 71–89.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Orion: Zero knowledge proof with linear prover time". *Annual International Cryptology Conference*. Springer. 2022, pp. 299–328.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. "BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes". *Proceedings of the 44th Annual International Cryptology Conference (CRYPTO '24)*. Vol. 14929. Lecture Notes in Computer Science. Springer, 2024, pp. 138–169. DOI: 10.1007/978-3-031-36832-3_6.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Martin Puškin,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose An Analysis of the WHIR Interactive Oracle Proof of Proximity, mille juhendaja on Helger Lipmaa, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Martin Puškin
11.02.2026