INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAX042

Thèse de doctorat

Efficient Protocols for Testing Proximity to Algebraic Codes

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le 16 juin 2022, par

SARAH BORDAGE

Composition du Jury :

Grégoire Lecerf Directeur de recherche, CNRS (LIX)	Président
Pierrick Gaudry Directeur de recherche, CNRS (LORIA)	Rapporteur
Swastik Kopparty Associate Professor, University of Toronto	Rapporteur
Eli Ben-Sasson Starkware Industries	Examinateur
Eleonora Guerrini Maîtresse de conférences, Université de Montpellier (LIRMM)	Examinatrice
Adeline Roux-Langlois Chargée de recherche, CNRS (IRISA)	Examinatrice
Gilles Zémor Professeur, Université de Bordeaux (IMB)	Examinateur
Daniel Augot Directeur de recherche, Inria Saclay	Directeur de thèse

Manuscrit de thèse

Sarah Bordage

VERSION PRÉ-SOUTENANCE

Contents

Introduction

In this thesis, we propose efficient protocols for solving the problem of testing proximity to algebraic linear codes. Such constructions can be viewed as solutions to the so-called low-degree testing problem, and variants thereof.

Our design approach is driven by one main motivation: the construction of *concretely efficient* proof systems for secure delegation of computation (also known as verifiable computing), and zero-knowledge proofs. Loosely speaking, proofs of computational integrity are short proofs that enable a verifier to probabilistically check the correct execution of a program, without having to run the computation again. For practical concerns, such proofs should not only be short, but also easy to generate and fast to verify. Besides, a zero-knowledge proof asserts that some statement is valid, without disclosing any meaningful information on the reasons why such a statement holds. Combining those two striking concepts leads to proofs of computational integrity that are extremely fast to verify and preserves the confidentiality of some sensitive inputs of the computation.

After several breakthroughs and tremendous progress, constructions that only a few years ago were barely seen as theoretical proofs of concept have finally led to genuine solutions for real-world applications and industrial deployments (proving once again that, after a few decades, fascinating but purely theoretical concepts may eventually find an interested audience in society).

There are several approaches for constructing succinct non-interactive arguments for verifying generic computations, with different tradeoffs in performance and security assumptions. Regarding long-term security, systems with no trusted setup and post-quantum security are the most desirable. For such systems, designing highly-efficient proximity tests for linear codes is crucial for practical implementations.

General context

Proof systems. In computational complexity theory, a computational problem is a task that can be solved by a computer. A computational problem can be thought as a collection of *instances*, each instance admitting a (possibly empty) set of *solutions*. A problem instance is the input of some computational problem, while the problem is the abstract question to be solved. Computational complexity theory classifies computational problems according to the amount of resources used to solve them (such as time, space or communication).

A *proof system* is a protocol in which an untrusted prover intends to convince a verifier with limited resources that a given statement (namely, an instance of a computational problem) is true by providing a supporting proof. Examples of such statements are "*the boolean circuit C has an assignment of its inputs that makes the output true*", "*the boolean formula φ is satisfiable*", or "*the graph G admits a 3-coloring*".

For instance, the **NP** class captures the set of theorems whose validity can be verified with full confidence in polynomial-time once a *proof* is provided (such a proof is often called a *certificate* or a *witness*). Closely related to the conjecture $\mathbf{P} \neq \mathbf{NP}$, it is generally thought that the solutions of problems are more difficult to find than to verify. An **NP** proof system has two natural basic requirements: any valid statement admits a valid proof that convinces a verifier, whereas there does not exist convincing proof for incorrect statements. The former requirement is called *completeness*,

while the latter is named *soundness*. The **NP** class imposes restrictions on the time needed to verify a proof (and no limitation on the time needed to generate it).

Interactive Proofs (IPs). Interactive proof systems [GMR85, Bab85] leverages the use of interaction and randomness for proving theorems. In this model, an all-powerful prover interacts over several rounds with a polynomial-time verifier which is allowed to use randomness. The interaction between the prover and the verifier is analogous to an "oral interview": a verifier gains trust about the honesty of the prover if the latter is able to provide convincing answers to randomly chosen questions. Compared to the class **NP**, the soundness requirement of an interactive proof system is relaxed to allow a probabilistic verifier to reject instances that are not in the language with high probability. The fundamental result $\mathbf{IP} = \mathbf{PSPACE}$ [LFKN90, Sha92] indicates that the use of interaction and randomness enable to efficiently verify solutions of larger class of problems, compared to the **NP** class. It is indeed commonly believed that $\mathbf{coNP} \neq \mathbf{NP}$, and thus $\mathbf{NP} \subsetneq \mathbf{PSPACE}$ since $\mathbf{coNP} \subseteq \mathbf{PSPACE}$.

When the verifier is allowed to send any kind of message and, in particular, may use private randomness, then the interactive proof system is *private-coin*. On the contrary, if the verifier's coin tosses are also known to the prover, the interactive proof system is said to be *public-coin* (or *Arthur-Merlin* type [Bab85]). Goldwasser and Sipser [GS86] showed that all languages with private-coin interactive proofs also have public-coin counterpart. The benefit of public-coin interactive protocols is that they can often be compiled into non-interactive proofs in the random oracle model, by using the Fiat-Shamir transformation [FS86].

Zero-knowledge proofs (ZKPs). Going back to the influential work of [GMR85], interactive proof systems were initially formalized with the motivation of introducing the puzzling possibility of proving a theorem without communicating any knowledge. Goldwasser, Micali, and Rackoff showed the existence of *zero-knowledge proofs*, which are proofs that do not reveal any information except the veracity of the prover's claim. A bit more formally, an interactive proof system is *zero-knowledge* if there exists an efficient probabilistic algorithm which, given as input the statement to be proved, outputs a transcript (of a simulated interaction) which is indistinguishable from the one produced by an actual interaction with a prover who knows the secret information.

Proof of knowledge. The soundness property can be strengthened by the concept of *knowledge soundness*, which yields to a *proof of knowledge*. Roughly speaking, a proof is a proof of knowledge if, for any prover that convinces a verifier that there exists a satisfying witness w for an instance x, it can be deduced than the prover actually *knows* such a witness. Specifically, a proof system $(\mathscr{P}, \mathscr{V})$ for an **NP** relation $\mathcal{R}$ is a *proof of knowledge with knowledge error* ε, if there exists a polynomial-time algorithm $\mathcal{E}$, called extractor, such that for every w and every prover $\widetilde{\mathscr{P}}$:

$$\Pr\left[(x,w) \in \mathcal{R} \mid w \leftarrow \mathcal{E}^{\widetilde{\mathscr{P}}}(x)\right] \geqslant \Pr[\langle \widetilde{\mathscr{P}}, \mathscr{V} \rangle(x) = 1] - \varepsilon,$$

where the notation $\mathcal{E}^{\widetilde{\mathscr{P}}}$ means that $\mathcal{E}$ gets black-box access to the algorithm $\widetilde{\mathscr{P}}$ with the ability to rewind $\widetilde{\mathscr{P}}$, and $\langle \widetilde{\mathscr{P}}, \mathscr{V} \rangle$ represents the output of $\mathscr{V}$ after interacting with $\widetilde{\mathscr{P}}$.

Probabilistically Checkable Proofs (PCPs). For some statement x and some proof π, we now consider a verifier which is bounded in the number r of coins tosses and the number of inspected bits

q of the proof π. Roughly speaking, a *probabilistically checkable proof* (PCP) is a special encoding of a nondeterministic witness w, which is robust in the sense that if there is even a very small fraction of error in the witness w, then errors will be spread throughout the proof π. Therefore, "proofs" of false statements that are encoded in this special format are guaranteed to have so many errors that it is possible to randomly read only a tiny portion of the proof π to assess its validity. Notice that, while error-correcting codes are widely known for their use in error detection and correction in data transmission over noisy communication channels, they can also encode computations in order to enable efficient program checking. The celebrated PCP theorem [AS92, ALM+98] gives another characterization of the **NP** class by saying that $\mathbf{NP} = \mathbf{PCP}[\log n, 1]$, *i.e.* all languages in **NP** can be decided with polynomial-size probabilistically checkable proofs, where the verifier tosses at most a logarithmic number of coins and queries a constant number of bits of the proof. The value of this constant is essentialy determined by the error probability the verifier is willing to tolerate. Since the verifier does not read the proof π in its entirety, such a proof π is called an *oracle proof*, and the verifier is said to have *oracle access* (*i.e.* query access) to it.

Succinct non-interactive arguments from PCPs. When the soundness of the proof system is only required to hold against computationally bounded (*e.g.* polynomial-time) prover, one gets *computational soundness*. Such a relaxation of soundness yields to *argument systems* [BCC88], instead of proof systems. In 1992, Kilian [Kil92] constructed a four-message interactive argument system for **NP**, based on collision-resistant hash functions and PCPs. The idea is to first require the prover to succinctly commit to a probabilistically checkable proof, and then to have the verifier requesting a partial opening of the commitment, at some randomly chosen location. The commitment needs to be short (whereas a probabilistically checkable proof is typically larger than the witness size), computationally binding (otherwise the prover could adaptively craft his answers to the verifier's queries), and must support local openings (again with low communication complexity). Based on collision-resistant hash functions, such a commitment can be implemented with Merkle trees. The commitment string is thus the root of the Merkle tree associated to the PCP, and a local opening of the commitment consists in providing a Merkle path, which has size logarithmic in the committed PCP length. Kilian's compiler enables *succinct* interactive argument for **NP**, in the sense that the communication complexity is polylogarithmic in the witness size. Specifically, the communication complexity and the verifier's running time of Kilian's protocol are bounded by $\mathsf{poly}(\kappa, |\mathsf{x}|, \log T)$, where κ is a security parameter, x is an instance of a **NP** language and T the time needed for the standard **NP** verification of a witness. Recently, [CMSZ21] showed that Kilian's protocol is post-quantum secure in the standard model, and thus yields to post-quantum succinct argument from any falsifiable assumption.

By combining Kilian's compiler with the Fiat-Shamir transformation [FS86], Micali [Mic95] showed how to construct succinct *non-interactive* arguments for **NP** in the random oracle model. Micali named such non-interactive arguments "Computationally sound proofs" but nowadays, the acronym SNARG (for "succinct non-interactive argument" [GW11]) is commonly used instead. The idea is to let the prover define the verifier's random challenges as the output of a cryptographic hash function applied to the instance and the commitment of the prover's message, considering that such functions are good enough approximation of random functions. Subsequent works showed that the transformation preserves both zero-knowledge and knowledge soundness [IMS12, MX13, Val08].

Nevertheless, the performance of PCP-based succinct arguments is constrained by the tremendous amount of computation needed to generate the PCP. Indeed, the complexity of the proof generation algorithm of known PCP constructions remains a massive bottleneck for practical implementations.

In parallel to PCP-based argument systems, the inefficiency of PCPs motivated the development of quite different techniques based on pairing-based cryptography in order to achieve concretely efficient SNARGs (both very short and very cheap to verify) [IKO07, Gro10, GGPR13, Lip13, BCI+13, PHGR13, BCG+13, Gro16] and deployed systems [Zca]. However, such constructions require a *trusted setup*, meaning that the initial parameters of the argument systems must be generated in a trusted way, since they contain secret randomness whose revelation would allow to forge convincing proofs of false statements. For real-world deployments, the realization of such a trusted setup implies a costly and logistically challenging "multiparty computation ceremony" [BCG+15].

The Interactive Oracle Proof model: a model designed for efficiency. One motivation for designing probabilistic proof systems with low communication complexity, fast generation and sublinear verification is the application to verifiable computation. Since the seminal works of Kilian [Kil92] and Micali [Mic95], a lot of efforts have been put into making PCPs efficient enough to obtain *practical* sublinear non-interactive arguments for delegating computation. In search of reducing the work required to generate such proofs, as well as the communication complexity of succinct arguments based on them, [BCS16, RRR16] introduced interactive oracle proof systems (IOPs), which generalize both PCPs, IPs and interactive PCPs [KR08]. An IOP can be viewed as a multi-round PCP where, in each round, the verifier sends some message and the prover answers with an oracle proof.

Extending the work of [Kil92, Mic00, Val08] and the Fiat-Shamir paradigm, Ben-Sasson, Chiesa and Spooner showed how to compile a public-coin interactive oracle proof of knowledge into a succinct non-interactive argument of knowledge (so-called SNARK) in the random oracle model [BCS16]. Chiesa, Manohar and Spooner showed that a SNARK constructed from an IOP with *round-by-round soundness* are unconditionally secure in the quantum random oracle model [CMS19].

Since the introduction of the IOP model, numerous works developed IOP constructions with efficiency parameters that outperform the best known PCP constructions [BCG+17, BBHR18a, BBHR19, KPV19, BCR+19, BCG+19, RR20, BCG20, BCL20, ZXZS20, RR21]. For instance, [BCG+17] constructed a linear-size constant-query IOP for circuit satisfiability, while it remains a fundamental open problem to build PCP for this problem with proof length linear in the witness size and constant query. The benefits of IOP-based SNARKs in comparison with SNARKs relying on the discrete-logarithm problem or pairing-based cryptography are three-fold: they can have a transparent setup (*i.e.* only public randomness is used), avoid cryptographic assumptions that are known to be broken by quantum attacks, and they can operate over fields that have smaller size and faster arithmetic (elliptic curve cryptography typically require 256-bit size field for 128 bits of security).

From a practical perspective, the IOP model enables the design of proof systems that are efficient enough to be deployed in the real world for concrete applications [BBHR19, Sta21a]. Yet, the prover running time remains a major bottleneck in IOP-based succinct arguments. A recent line of works focuses on constructing IOPs with linear-time prover [BCG20, BCL20, GLS+21, RR21]. Designing practical and specific tools for those constructions in order to replace some generic components that are theoretical in nature (such as the Mie's PCP of Proximity for nondeterministic language [Mie09]) would probably enable to construct concretely efficient SNARKs from IOPs with linear prover running time.

Arithmetization: from computational problems to low-degree testing

Most PCP and IOP constructions share the same design principles. Roughly speaking, the prover provides an encoding of the computation, using some error-correcting code. Such a code is chosen so that there exists a proximity test for it, namely a way for a verifier to check that an alleged codeword indeed belongs to the code (this can be done by using locally testable codes, or by asking the prover to provide an auxiliary proof of proximity). Additionally, the PCP or IOP allows some probabilistic verification of the fact that the computation encoded by the prover is a correct computation.

In the context of proof systems, the generic term of *arithmetization* [LFKN90] refers to a class of algebraic techniques that enable to reduce a computational problem to an algebraic one. Such an algebraic problem typically involves a system of polynomial equations where polynomials are defined over some finite field, and have degree significantly smaller than the field size (so-called *low-degree polynomials*). In other words, an instance-witness pair (x, w) belongs to a given binary relation $\mathcal{R}$ if and only if certain polynomial identities hold. The validity of such identities are then probabilistically verified using various techniques, for instance by checking that a certain polynomial vanishes on a given subset, or testing whether the sum of its evaluations on a given domain is equal to zero. The latter problem is solved by the renowned (multivariate) "sumcheck protocol" of [LFKN90], or a univariate variant of it [BCR+19].

Arithmetization techniques for constructing proof systems emerged from the study of interactive proofs [Bab85, GMR85] and have been enhanced and fruitfully applied to other broad families of proof systems, including multi-prover interactive proofs [BGKW88, BFL90], probabilistically checkable proofs [BFLS91, AS92, ALM+98], interactive oracle proofs [RRR16, BCS16] and zero-knowledge proof systems [GMR85]. Arithmetization techniques generally involve *algebraic linear codes* (but not always [BCG20, BCL20]), such as polynomial codes (for instance Reed-Solomon codes and Reed-Muller codes) or generalization of them (the so-called algebraic geometry codes).

Many constructions of PCPs, including the ones used to prove the PCP theorem [AS92, ALM+98], rely on multivariate polynomials, and thus implies the following problem known as *multivariate low-degree testing*. Given a proximity parameter $\delta \in (0,1)$, a degree bound d (typically much smaller than q) and oracle access to a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$, determine with a few queries whether f is a polynomial function of degree less than d, or δ-far in relative Hamming distance from being low-degree. Such low-degree tests correspond to proximity tests for codes corresponding to evaluations of multivariate polynomials of bounded degree. Low-degree tests have been the subject of a substantial body of research during the past four decades, which was initially motivated by their relations with constructions of probabilistic proof systems.

More generally, at the end of arithmetization, an instance x of a computational problem (such as circuit satisfiability) is mapped to one or several algebraic codes $C_1, C_2, \ldots, C_n$, together with a set of constraints that are to be satisfied. Loosely speaking, arithmetization ensures that there exists a witness w that satisfies the instance x if and only if there exit some codewords of $C_1, C_2, \ldots, C_n$ that satisfy the set of constraints. In particular, if x is not a satifiable instance, then any sequence of words $w_1, w_2, \ldots, w_n$ such that w_i is sufficiently close to C_i for $i \in \{1, 2, \ldots, n\}$ will fail to satisfy the prescribed constraints. Therefore, the verification procedure typically consists in testing whether words sent by the prover are in close proximity to the given algebraic linear codes, and to randomly check that they satisfy the prescribed constraints.

As a result, the proximity problem for algebraic linear codes plays a central role in constructions of PCP (and IOP) systems. For an error-correcting code C, such a problem can be roughly formulated

as follows: "Given a code $C \subset \Sigma^D$ and oracle access to a function $f : D \to \Sigma$, the goal is to determine whether $f \in C$ and f is far from any codeword of C."

Reed-Solomon proximity testing. In [BBHR18a], the authors mentioned that a significant bottleneck of PCP-based proof systems regarding proof generation and communication complexity stems from the inefficiency of the solutions to the multivariate low-degree testing problem. Ben-Sasson and Sudan [BS08] considered for the first time univariate polynomials instead of multivariate ones, which leads to the breakthrough result of PCP construction with quasilinear proof length and constant query complexity [BS08, Din07]. This result gives the constant-query PCP of shortest length known to date. More recently, efficient transparent and zero-knowledge non-interactive arguments have been designed from interactive variants of PCPs, by successfully relying on Reed-Solomon (RS) codes [AHIV17, BBHR19, BCR+19, BCG+19, KPV19, COS20, ZXZS20]. Those succinct arguments crucially require an efficient proximity test for Reed-Solomon codes.

Building upon [PS94, BS08], Ben-Sasson, Bentov, Horesh and Riabzev [BBHR18a] constructed a prover-efficient IOP of Proximity (IOPP) for testing proximity to Reed-Solomon codes evaluated over well-chosen evaluation points and named it FRI protocol (this IOPP was further improved in [BKS18, BGKS20, BCI+20]). The FRI protocol admits linear prover time, logarithmic verifier time and logarithmic query complexity. This protocol is sub-optimal for some parameters but highly-efficient in practice, which makes it a crucial component of systems deployed in the real-world [BBHR19, Sta, Sta21b]. Indeed, short IOPPs with constant-query for Reed-Solomon codes do exist [BCG+17, RR20], but such proximity tests are theoretical in nature.

At the cost of a larger query complexity (*i.e.* logarithmic), the FRI protocol is ingeniously simple to implement. The tasks of the prover and the verifier consist only in computing univariate polynomial interpolations of small degree (*e.g.* 2 or 4) and performing cryptographic hashing. On the downside, known proximity tests for Reed-Solomon codes require the field to be larger than the block length of the code, and to admit additive or multiplicative subgroups of large smooth order[1] [BS08, BBHR18a].

Main contributions

In this thesis, we study the problem of testing proximity to several families of error-correcting codes. We give practical and concretely efficient solutions to solve a proximity testing problem for a code C, *i.e.* distinguishing between the case where an input word, given as an oracle, belongs to C and the one where it is far from every codeword of C.

We propose solutions to this problem in the Interactive Oracle Proof model [BCS16], which has demonstrated to be particularly promising for the design of proof systems in the past few years. Our constructions are based on the FRI protocol for Reed-Solomon proximity testing [BBHR18a]. Inspired from [BBHR18a], we formulate and analyze an abstract framework for constructing efficient IOPs of Proximity for linear codes. This approach allows us to provide efficient protocols for linear *algebraic* codes that are relevant for constructing efficient proof systems: linear codes that are defined from evaluations of bounded degree multivariate polynomials on product sets and rational functions on

[1] At the time of writing, authors of [BCKL21] advertise for a proximity test for Reed-Solomon codes over any finite field, which may mitigate this issue in the future. Besides, the cited work provides an elegant FFT-like algorithm running in time $O(n \log n)$ over any finite field, but implemented code and concrete performance are not yet provided.

algebraic curves. Although we do not propose any implementation for our protocols, we confidently state that our proximity tests perform very well in practice. Their design is indeed very similar in nature to the highly-efficient FRI protocol, and only require univariate polynomial interpolations with very small degrees.

Proximity tests for multivariate polynomial codes

This contribution is a joint work with Daniel Augot and Jade Nardi.

Multivariate low-degree tests fall into two flavours, depending on whether one requires a bound on the total degree or the degree in each variable. In the former case, the low-degree test can be viewed as a proximity test for Reed-Muller codes. In the latter case, it corresponds to a proximity test to a tensor product of Reed-Solomon codes. While multivariate low degree tests have been extensively studied due to their role in the constructions of proof systems, they have not been the subject of any specific construction in the interactive oracle proof model.

We develop interactive oracle proofs of proximity (IOPP) for tensor products of Reed-Solomon codes and for Reed-Muller codes with efficiency parameters that compared well with those of the IOPP for Reed-Solomon codes of [BBHR18a]. Counted in field operations, we construct IOPP with linear prover running time and logarithmic verification time (with respect to the block length of the code).

More specifically, the treatment of the total degree case requires some technical precautions, which result in a small loss in efficiency compared to the univariate case. For the interesting regime where the number of indeterminates m is assumed to be a small constant independent of the block length, the efficiency loss is minor. In contrast, for tensor product of Reed-Solomon codes of block length N, we construct an IOPP with strictly linear-size prover and proof length, and strictly logarithmic query and verifier complexities even when m is not a constant. Most notably, the actual constants involved in the complexities of our IOPP are the same as for the univariate case (solved by the FRI protocol [BBHR18a]).

Proximity tests for algebraic geometry codes

This work is the result of an initial joint work with Jade Nardi, and a subsequent collaboration with Matthieu Lhotel and Hugues Randriambololona.

An algebraic geometry (AG) code is a vector space formed by evaluations on a set of rational points of an algebraic variety $\mathcal{P} \subset \mathcal{X}$ of functions in the Riemann-Roch space $L_{\mathcal{X}}(D)$.

We initiate the study of proximity testing to Algebraic Geometry (AG) codes. An AG code $C = C(\mathcal{X}, \mathcal{P}, D)$ is a vector space associated to evaluations on $\mathcal{P}$ of functions in the Riemann-Roch space $L_{\mathcal{X}}(D)$. A couple of works [BKK+13, BCG+17] constructed proof systems based on tensor product of AG codes with constant-size alphabet [GS95]. In [BKK+13, BCG+17], proximity testing is made possible thanks to the local properties of tensor products of codes. Prior to our work, there was no efficient proximity tests for plain AG codes.

We construct an Interactive Oracle Proof of Proximity (IOPP) for some families of AG codes by generalizing the IOPP for Reed-Solomon codes of [BBHR18a]. We identify sufficient conditions for designing efficient IOPP systems for AG codes.

Our approach relies on a neat decomposition of the Riemann-Roch space of any invariant divisor under a group action on a curve into several explicit Riemann-Roch spaces on the quotient curve. Along the way, we provide a framework in which a proximity test to C can be reduced to one to a shorter code C'. Iterating this process thoroughly, we end up with a membership test to a code with significantly smaller length.

In addition to proposing the first proximity test targeting AG codes, we achieve parameters that also compete well with the FRI protocol. Specifically, we study AG codes on Kummer curves and curves in the Hermitian tower. Notably, the latter families of AG codes can be defined over polylogarithmic-size alphabet and require less algebraic constraints on the underlying finite field, compared to Reed-Solomon codes. We specialize our generic AG-IOPP construction to these two specific subfamilies of AG codes, reaching a linear prover running time and logarithmic verification on Kummer curves, and quasilinear prover time with polylogarithmic verification for the Hermitian tower.

Organization of the manuscript

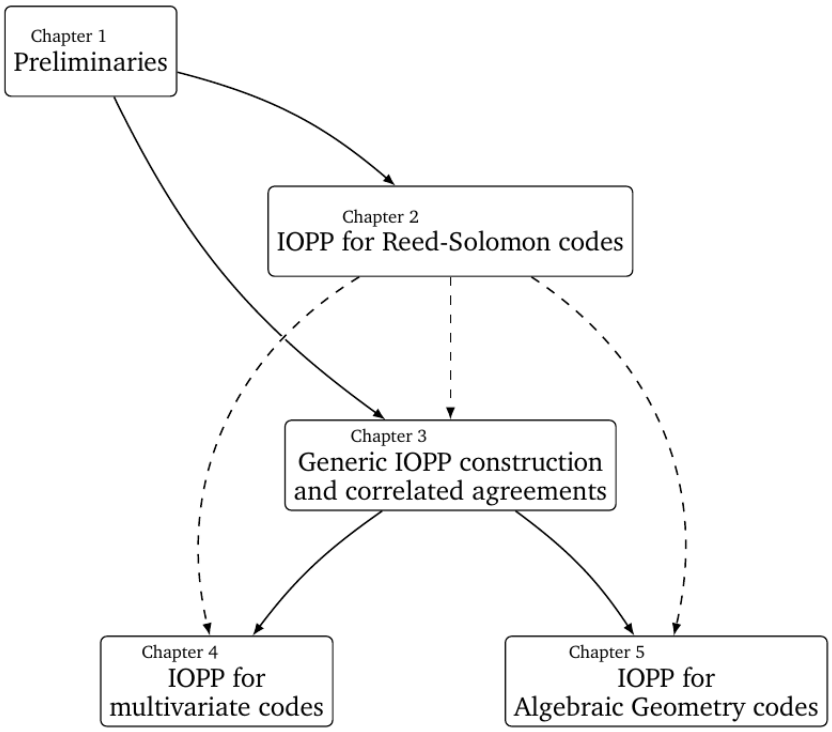

Figure 1: Outline of the thesis

CHAPTER 1

Background material

1.1 Notations and conventions used throughout the manuscript

Finite fields. Every field considered in this manuscript is finite. We denote by $\mathbb{F}$ a finite field and by $\mathbb{F}_q$ the finite field of size q. The algebraic closure of a finite field $\mathbb{F}$ is denoted by $\bar{\mathbb{F}}$.

We say that L is a subgroup in $\mathbb{F}$ if it is either a subgroup of $(\mathbb{F} \setminus \{0\}, \times)$ or $(\mathbb{F}, +)$. We use the notation $\mathbb{F}_q^\times$ for group of invertible elements of $\mathbb{F}_q$. The multiplicative subgroup generated by an element $g \in \mathbb{F}_q^\times$ will be denoted by $\langle g \rangle$.

Intervals. A (real) interval is denoted with square brackets when endpoints are included. A parenthesis replaces a square bracket when a endpoint is excluded. For two integers a, b with $a < b$, we use the notation $[\![a, b]\!]$ for the set of integers $\{a, a+1, \ldots, b\}$. For an integer $n > 1$, we use the shorthand notation $[n]$ to refer to $[\![1, n]\!]$.

Tuples and sets. The indicator function of a set S is denoted by $\mathbb{1}_S$. For two finite sets A, B, we use the power set notation B^A for the set of maps with domain A and range B. Given an alphabet Σ, we denote the set of all finite strings over Σ by Σ^*, and by Σ^n the set of strings of length n. We identify Σ^n with $\Sigma^{[n]}$, by viewing $\boldsymbol{u} \in \Sigma^n$ as the evaluation tuple of the function $f : \{1, 2, \ldots, n\} \to \Sigma$ such that $\boldsymbol{u} = (u_1, \ldots, u_n) = (f(i))_{i \in [n]}$.

The set of integers is denoted by $\mathbb{Z}$, and the set of non-negative integers by $\mathbb{N}$. Tuples are written in bold lower-case letters. For $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{F}_q^m$ and $\boldsymbol{u} = (u_1, \ldots, u_m) \in \mathbb{N}^m$, the notation $\boldsymbol{x}^{\boldsymbol{u}}$ represents the product $\boldsymbol{x}^{\boldsymbol{u}} \coloneqq x_1^{u_1} \cdots x_m^{u_m}$. The Hamming weight of a vector $\boldsymbol{x} \in \Sigma^n$ is denoted by $\mathsf{w}_\mathsf{H}(\boldsymbol{x})$.

Polynomials over finite fields. For indeterminates $X_1, \ldots, X_m$, we use a bold capital letter $\boldsymbol{X}$ as a shorthand for $\boldsymbol{X} = (X_1, \ldots, X_m)$ when the number of indeterminates is clear from context. Accordingly, we denote by $\mathbb{F}_q[\boldsymbol{X}]$ the ring of m-variate polynomials $\mathbb{F}_q[X_1, \ldots, X_m]$. For a multivariate polynomial $P \in \mathbb{F}_q[\boldsymbol{X}]$, we denote by $\deg P$ the total degree of P, and $\deg_{X_i} P$ the degree of P with respect to the indeterminate X_i. We say that P has individual degree bounded by d if its degree in each variable X_i is less than d.

We denote by $\mathbb{F}_q[X]_{<k}$ the set of univariate polynomials with coefficients in $\mathbb{F}_q$ and degree less than k.

Evaluations of polynomials. Given a function $f : D \to \mathbb{F}_q$ evaluated over some finite domain $D \subseteq \mathbb{F}_q$, we will denote by a "hatted" letter the interpolant polynomial of f, i.e. $\hat{f} \in \mathbb{F}_q[X]$ is the polynomial of minimal degree such that for all $x \in D, \hat{f}(x) = f(x)$. Similarly, for a multivariate function $f : D_1 \times D_2 \times \cdots \times D_m \to \mathbb{F}$, we use an hatted letter $\hat{f}$ to denote a polynomial $\hat{f} \in \mathbb{F}[\boldsymbol{X}]$

such that, for all $\boldsymbol{x} \in D_1 \times D_2 \times \cdots \times D_m, f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x})$. For a set $D \subset \mathbb{F}_q$ and a polynomial $P \in \mathbb{F}_q[X]$ of degree $\deg P < |D|$, we slightly abuse notations and denote by $P_{|D} : D \to \mathbb{F}_q$ the restriction to domain D of the polynomial function corresponding to $P \in \mathbb{F}_q[X]$.

Languages and relations. A language $\mathcal{L}$ is a set of strings $\mathcal{L} \subseteq \Sigma^*$ over some alphabet Σ. Examples of such alphabets are $\Sigma = \{0,1\}$, or $\Sigma = \mathbb{F}$ for some finite field $\mathbb{F}$. A binary relation $\mathcal{R}$ is a subset of a cartesian product $\mathcal{X} \times \mathcal{W}$, where $\mathcal{X}, \mathcal{W} \subseteq \Sigma^*$ for some alphabet Σ. Usually, elements $\mathsf{x} \in \mathcal{X}$ are called *instances* and elements of $\mathsf{w} \in \mathcal{W}$ are called *witnesses*. For a relation $\mathcal{R}$, we assume that the size of w is bounded by some (computable) function of the size of x. The language associated to a binary relation $\mathcal{R}$ is

$$\mathcal{L}(\mathcal{R}) \coloneqq \{\mathsf{x} \in \mathcal{X} \mid \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}.$$

For $\mathsf{x} \in \mathcal{X}$, we denote by $\mathcal{R}_{|\mathsf{x}}$ the (possibly empty) set of witnesses corresponding to x, i.e.

$$\mathcal{R}_{|\mathsf{x}} \coloneqq \{\mathsf{w} \in \mathcal{W} \mid (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}.$$

Probabilities. For a finite set S, we denote by $\Pr_{x \in S}[E(x)]$ the probability that the random event $E(x)$ occurs when x is sampled uniformly at random from S. The expectation of a random variable X is denoted by $\mathbf{E}[X]$. When we write $x \xleftarrow{\$} S$, we mean that x is sampled uniformly at random from S.

Distances and agreements. We denote by $\Delta_\Sigma : \Sigma^n \times \Sigma^n \to [0,1]$ the relative Hamming distance over Σ, i.e. the ratio of coordinates on which they differ. The subscript will be omitted when the alphabet Σ is clear from context.

For some $\delta \in [0,1]$ and some $\boldsymbol{u}, \boldsymbol{u}' \in \Sigma^n$, we say that $\boldsymbol{u}$ is δ-far from $\boldsymbol{u}'$ when $\Delta(\boldsymbol{u}, \boldsymbol{u}') > \delta$. Otherwise, we say that $\boldsymbol{u}$ is δ-close to $\boldsymbol{u}'$. For a set $S \subseteq \Sigma^n$ and $\boldsymbol{u} \in \Sigma^n$, we define $\Delta(\boldsymbol{u}, S)$ as the minimum over $\boldsymbol{s} \in S$ of the distances $\Delta(\boldsymbol{u}, \boldsymbol{s})$. If S is empty, we use the convention that any string is at relative distance 1 of S. We say that $\boldsymbol{u}$ is δ-far from S if $\Delta(\boldsymbol{u}, S) > \delta$. Otherwise, we say that $\boldsymbol{u}$ is δ-close to S.

The relative agreement between $\boldsymbol{u}, \boldsymbol{u}' \in \Sigma^n$, denoted by $\mathsf{agree}(\boldsymbol{u}, \boldsymbol{u}')$, is equal to $\mathsf{agree}(\boldsymbol{u}, \boldsymbol{u}') = 1 - \Delta(\boldsymbol{u}, \boldsymbol{u}')$. It is the ratio of coordinates on which $\boldsymbol{u}$ and $\boldsymbol{u}'$ coincide. We can extend this notion to any set S, by setting $\mathsf{agree}(\boldsymbol{u}, S) = 1 - \Delta(\boldsymbol{u}, S)$.

Asymptotic notations. Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be two functions. We write $f(n) = O(g(n))$ is there is a constant c such that, for every sufficiently large n, we have $f(n) \leqslant cg(n)$. We say that $f(n) = o(g(n))$ if for every $\varepsilon > 0$ and every sufficiently large n, $f(n) \leqslant \varepsilon g(n)$. If $g(n) = O(f(n))$, we write $f(n) = \Omega(g(n))$. If both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ hold, then we write $f(n) = \Theta(g(n))$.

The notation $f(n) = O_c(g(n))$ means that c is treated as a constant independent of n. We write $f(n) = \widetilde{O}(g(n))$ if $f(n) = O\left(g(n) \log^c g(n)\right)$ for some constant $c > 0$.

We write $f(n) = \mathsf{poly}(n)$ if $f(n) = n^{O(1)}$, and $f(n) = \mathrm{polylog}(n)$ if $f(n) = O(\log^c n)$ for some constant $c > 0$.

Logarithms. We denote by $\log_b$ the logarithm to base b. By default, we write log without subscript to refer to the logarithm to base 2.

1.2 Proof systems and cryptographic arguments

For an introduction to standard notions of computational complexity theory, we refer to [AB09, Gol08]. We also refer the reader to [Tha22] for a recent survey of constructions of proof systems and succinct arguments.

1.2.1 Nondeterministic languages

The **NP** class can be related to the familiar notion of a *mathematical proof*. The intuition is that a proof can be very hard to find but much easier to verify. An **NP** verifier is an algorithm receiving the statement of a theorem and a purported proof. The completeness and soundness requirements come naturally: any valid statement can be proved, whereas a false theorem does not admit a correct proof. Originally, the **NP** class was defined as the set of languages that can be decided by a *nondeterministic polynomial-time* Turing machine, namely a machine making a non-deterministic choice (a "guess") about which state to transition to. We adopt an equivalent definition, which highlights the fact that the complexity class **NP** captures the set of problems whose solutions can be verified in polynomial-time.

Definition 1.1 (**NP** class)**.** *A language $\mathcal{L}$ is in* **NP** *if there exists a polynomial-time deterministic algorithm $\mathcal{V}$ such that the following conditions hold.*

Completeness: *For every instance $\mathsf{x} \in \mathcal{L}$, there exists a witness w of $\mathsf{poly}(|\mathsf{x}|)$-size such that the verifier $\mathcal{V}$ accepts, i.e. $\mathcal{V}(\mathsf{x}, \mathsf{w}) = 1$.*

Soundness: *For every instance $\mathsf{x} \notin \mathcal{L}$ and every string $\tilde{\mathsf{w}}$ of $\mathsf{poly}(|\mathsf{x}|)$-size, the verifier $\mathcal{V}$ rejects, i.e. $\mathcal{V}(\mathsf{x}, \tilde{\mathsf{w}}) = 0$.*

The foregoing definition can be generalized to verifier running in time $O(T(n))$.

Definition 1.2. *A language $\mathcal{L}$ is in the complexity class* **NTIME**$(T(n))$ *if there exists a deterministic algorithm $\mathcal{V}$ running in time $O(T(n))$ such that the following conditions hold.*

Completeness: *For every instance $\mathsf{x} \in \mathcal{L}$ of size n, there exists a witness w of size $O(T(n))$ such that the verifier $\mathcal{V}$ accepts, i.e. $\mathcal{V}(\mathsf{x}, \mathsf{w}) = 1$.*

Soundness: *For every instance $\mathsf{x} \notin \mathcal{L}$ and every string $\tilde{\mathsf{w}}$ of size $O(T(n))$, the verifier $\mathcal{V}$ rejects, i.e. $\mathcal{V}(\mathsf{x}, \tilde{\mathsf{w}}) = 0$.*

In particular, the space complexity of the verifier cannot exceed $O(T(n))$. The complexity classes **NP** and **NEXP** can be defined in terms of **NTIME** as follows:

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(n^k) \quad \text{and} \quad \mathbf{NEXP} = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(2^{n^k}).$$

Let $\mathcal{V}$ be an algorithm deciding a nondeterministic language $\mathcal{L} \in \mathbf{NTIME}(T(n))$. Then there is a relation $\mathcal{R}_{\mathcal{V}}$ induced by $\mathcal{L}$, which is $\mathcal{R}_{\mathcal{V}} \coloneqq \{(\mathsf{x}, \mathsf{w}) \mid \mathcal{V}(\mathsf{x}, \mathsf{w}) = 1\}$. The algorithm $\mathcal{V}$ is generally assumed to be known and fixed, and we simply say that $\mathcal{R} = \mathcal{R}_{\mathcal{V}}$ is the relation induced by $\mathcal{L}$.

1.2.2 Probabilistic oracle machines

We recall that a probabilistic (Turing) machine is an "extension" of a Turing machine, where the machine "chooses" its next move uniformly at random among a finite set of possible states. We always assume that a (Turing) machine outputs either "1" (for "accept") or "0" (for "reject").

Without loss of generality, one can view such a machine as tossing unbiased coins. The output of a probabilistic machine M on input x is a random variable defined over the probability space of all possible coin tosses of M (the *internal randomness* of M). The probability that M outputs $b \in \{0,1\}$ on input x is denoted by

$$\Pr\left[M(x) = b\right].$$

Equivalently, one can view a probabilistic machine as a deterministic machine which receives an auxiliary random input r, in addition to its ordinary input x. In that case, the probability that M outputs $b \in \{0,1\}$ on input x and randomness r is denoted by

$$\Pr_r\left[M(x;r) = b\right].$$

A *probabilistic oracle machine* is a probabilistic machine that has access to a black-box, known as *oracle*. In this thesis, we consider *oracle functions*, namely oracles that computes a certain function: an oracle function $f : D \to \Sigma$ is an oracle receiving some query $x \in D$ and responds with $f(x)$. We recall that any string $\boldsymbol{u} \in \Sigma^\ell$ of length $\ell \in \mathbb{N}$ can be viewed as a function $\boldsymbol{u} : [\ell] \to \Sigma$.

Let $\mathscr{V}$ be a probabilistic oracle machine. We denote by $\mathscr{V}^f(x)$ the output of $\mathscr{V}$ on input x and with oracle access to an oracle function f. The probability that $\mathscr{V}$ outputs $b \in \{0,1\}$ is denoted by

$$\Pr\left[\mathscr{V}^f(x) = b\right],$$

where the probability is taken over the internal randomness of $\mathscr{V}$. If the queries of a probabilistic oracle machine $\mathscr{V}$ are only determined by its explicit input and its internal randomness, we say that $\mathscr{V}$ is non-adaptive. In particular, queries are independent of the previous oracle answers.

1.2.3 Probabilistically checkable proofs

We recall the definition of a probabilistically checkable proof system (PCP) [BFLS91, AS92, ALM+98] for a language $\mathcal{L}$.

Definition 1.3 (Probabilistically Checkable Proof (PCP))**.** *Let $\mathcal{L} \subseteq \Sigma^*$ be a language and $\mathsf{q}, \mathsf{r} : \mathbb{N} \to \mathbb{N}$. A* probabilistically checkable proof system *(PCP) for $\mathcal{L}$ consists of a probabilistic (non-adaptive) polynomial-time oracle machine $\mathscr{V}$ (called* PCP verifier*) satisfying:*

Efficiency: *On input $\mathsf{x} \in \Sigma^n$ and given oracle access to a proof string π of length at most $2^{\mathsf{k}(n)}\mathsf{q}(n)$, $\mathscr{V}$ uses at most $\mathsf{k}(n)$ random coins and makes at most $\mathsf{q}(n)$ non-adaptive queries to π.*

Perfect completeness: *For every $\mathsf{x} \in \mathcal{L}$, there exists a proof π such that $\mathscr{V}$ accepts with probability 1 over its internal randomness, i.e.*

$$\Pr\left[\mathscr{V}^\pi(\mathsf{x}) = 1\right] = 1.$$

Soundness: *For every $\mathsf{x} \notin \mathcal{L}$ and every proof $\tilde{\pi}$, $\mathscr{V}$ accepts with probability bounded by $\frac{1}{2}$ over its internal randomness, i.e.*

$$\Pr\left[\mathscr{V}^{\tilde{\pi}}(\mathsf{x}) = 1\right] \leqslant \frac{1}{2}.$$

A language $\mathcal{L}$ is in the complexity class $\mathbf{PCP}[\mathsf{r}(n), \mathsf{q}(n)]$ *if there is a PCP verifier $\mathcal{V}$ for $\mathcal{L}$ that uses $O(\mathsf{r}(n))$ random coins and makes $O(\mathsf{q}(n))$ queries.*

In contrast to an NP verifier, a PCP verifier is allowed to use randomness, but is restricted to examine a small portion of the purported proof. Compared to Definition 1.1, the soundness requirement is relaxed, in the sense that the verifier is only expected to reject instances that are not in the language with some probability. Loosely speaking, a probabilistic checkable proof π for an instance $\mathsf{x} \in \mathcal{L}$ is a robust encoding of a nondeterministic witness w for x that allows a PCP verifier to randomly read a small fraction of symbols of π, and decide with high probability whether the proof is valid or not. Such probabilistically checkable proof π is at least as long as a witness w, but is not explicitly given as input to the PCP verifier.

Notice that we have

$$\mathbf{PCP}[\mathsf{r}(n), \mathsf{q}(n)] \subseteq \mathbf{NTIME}(2^{O(r(n))q(n)}),$$

since any nondeterministic machine can guess a correct proof in time $2^{O(r(n))}q(n)$ and deterministically verify it by running a PCP verifier for all $2^{O(r(n))}$ possible choices of random coin tosses. Therefore,

$$\mathbf{PCP}(\log n, 1) \subseteq \mathbf{NTIME}(2^{O(\log n)}) = \mathbf{NP}.$$

The celebrated PCP theorem states that the reverse inclusion also holds.

Theorem 1.4 ([AS92, ALM+98])**.** $\mathbf{NP} = \mathbf{PCP}[\log n, 1]$.

A PCP can be strengthened with the notion of *proximity*. A PCP of Proximity (also known as *assignment tester*) [DR04, BGH+04] for a binary relation $\mathcal{R}$ consists in probabilistic verifier having oracle access to both a purported witness w for some instance x, and an oracle proof π.

Definition 1.5 (PCP of Proximity (PCPP))**.** *A PCP of Proximity for a binary relation $\mathcal{R}$ with proximity parameter $\delta : \mathbb{N} \to [0,1]$ and soundness error $\varepsilon : \mathbb{N} \to [0,1]$ is a probabilistic (non-adaptive) polynomial-time machine $\mathcal{V}$ satisfying the following two conditions (where probabilities are taken over the internal randomness of $\mathcal{V}$).*

Completeness: *For every $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, there exists an oracle proof $\pi \in \Sigma^*$ such that $\mathcal{V}$ always accepts when given as input x and oracle access to (w, π), i.e.*

$$\Pr\left[\mathcal{V}^{\mathsf{w},\pi}(\mathsf{x}) = 1\right] = 1.$$

Soundness: *For every pair (x, w) such that $\Delta(\mathsf{w}, \mathcal{R}_{|\mathsf{x}}) > \delta(n)$ and every proof $\tilde{\pi} \in \Sigma^*$, the verifier $\mathcal{V}$ accepts when given as input x and oracle access to $(\mathsf{w}, \tilde{\pi})$ with probability at most $\varepsilon(n)$, i.e.*

$$\Pr\left[\mathcal{V}^{\mathsf{w},\tilde{\pi}}(\mathsf{x}) = 1\right] \leqslant \varepsilon(n).$$

1.2.4 Interactive oracle proofs

Interactive oracle proof systems (IOPs) [BCS16, RRR16] are information-theoretic proof systems generalizing interactive proof systems (IPs) [GMR85], probabilistically checkable proof systems (PCPs) [BFLS91, AS92, ALM+98] and interactive probabilistically checkable proof systems (IPCPs) [KR08].

Definition 1.6 (Interactive Oracle Proof (IOP))**.** *Let $\mathscr{P}$ and $\mathscr{V}$ be two probabilistic algorithms where $\mathscr{P}$ has unbounded running time and $\mathscr{V}$ is a probabilistic (non-adaptive) polynomial time machine. We say that $(\mathscr{P}, \mathscr{V})$ is a* r*-round (public-coin)* interactive oracle proof system (IOP) *for a binary relation $\mathcal{R}$ with soundness error $\varepsilon : \mathbb{N} \to [0,1]$ if the following properties are satisfied (where probabilities are taken over the internal randomness of $\mathscr{V}$).*

Interaction phase: *The prover $\mathscr{P}$ and the verifier $\mathscr{V}$ receive as input an instance* x*, and the prover is additionally given some string* w*. The prover $\mathscr{P}$ and $\mathscr{V}$ interact over* r *round as follows. At each round, the verifier sends a message m_i chosen uniformly at random, and the prover answers with an oracle message π_i to the verifier.*

Query phase: *At the end of the interaction, the verifier queries the oracle $\pi_1, \ldots, \pi_{\mathsf{r}}$ sent by the prover. Query locations are generated using public randomness. The verifier outputs either "1" for "accept" or "0" for "reject". We denote by $\langle \mathscr{P}, \mathscr{V} \rangle$ the random variable representing the decision of the verifier after interacting with a prover $\mathscr{P}$.*

Perfect completeness: *If $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, then* $\Pr\left[\langle \mathscr{P}(\mathsf{x}, \mathsf{w}), \mathscr{V}(\mathsf{x}) \rangle = 1\right] = 1$.

Soundness: *If $\mathsf{x} \notin \mathcal{L}(\mathcal{R})$, then for any unbounded malicious prover $\widetilde{\mathscr{P}}$,* $\Pr\left[\langle \widetilde{\mathscr{P}}, \mathscr{V}(\mathsf{x}) \rangle = 1\right] \leqslant \varepsilon(n)$*, where n denotes the size of* x*.*

Efficiency measures. Beyond soundness error, we are interested in the following efficiency measures:

- the *alphabet* $\mathsf{a}(n)$ of the IOP, *i.e.* the alphabet of the prover's messages;
- the *round complexity* $\mathsf{r}(n)$, namely the number of rounds of interaction;
- the *proof length* $\mathsf{l}(n)$, which is the sum of the sizes of the messages sent by the prover, counted in number of symbols of the alphabet ;
- the *query complexity* $\mathsf{q}(n)$, namely the total number of queries made by the verifier to the prover's messages;
- the *prover complexity* $\mathsf{tp}(n)$, which is the time spent by the prover $\mathscr{P}$ to generate all its messages;
- the *verifier complexity* $\mathsf{tv}(n)$, which is the running time of the verifier to output a decision, when queries and query-answers are given as inputs.

The IOP alphabet will often be a finite field $\mathbb{F}$. In that case, and unless specified otherwise, we state complexities counted in field operations, assuming that each field operation has a constant cost.

On the isolation of interactive and query phases. As mentioned in [BCR+19], the verifier $\mathscr{V}$ of a public-coin IOP system $(\mathscr{P}, \mathscr{V})$ can be thought as a couple of algorithms $(\mathscr{V}_i, \mathscr{V}_q)$ where:

- $\mathscr{V}_i$ outputs messages $m_1, \ldots, m_r$ sampled uniformly at random (during the interaction phase);
- $\mathscr{V}_q$ is a probabilistic oracle machine which outputs the decision of $\mathscr{V}$ (during the query phase).

Each of the algorithm $\mathscr{V}_i$ and $\mathscr{V}_q$ receives as auxiliary input the necessary randomness. Then the verifier $\mathscr{V}$ accepts with soundness error $\varepsilon \leqslant \varepsilon_i + \varepsilon_q$, where ε_i and ε_q satisfy

$$\Pr_{r_i}\left[\Pr_{r_q}\left[\mathscr{V}_q^{\pi_1,\ldots,\pi_{\mathsf{r}}}(\mathsf{x}, m_1, \ldots, m_{\mathsf{r}}; r_q) = 1 \geqslant \varepsilon_q\right] \,\middle|\, \begin{array}{r} (m_1, \ldots, m_{\mathsf{r}}) \leftarrow \mathscr{V}_i(\mathsf{x}; r_i) \\ (\pi_1, \ldots, \pi_{\mathsf{r}}) \leftarrow \widetilde{\mathscr{P}}(m_1, \ldots, m_{\mathsf{r}}) \end{array}\right] \leqslant \varepsilon_i.$$

for all $x \notin \mathcal{L}(\mathcal{R})$ and malicious prover $\widetilde{\mathcal{P}}$. By repeating α times the (non-interactive) query phase of an IOP system $(\mathcal{P}, \mathcal{V})$ with fresh randomness, one can get an IOP with soundness error $\varepsilon_i + \varepsilon_q^\alpha$, query complexity αq and verifier complexity αtv. Proof length, round complexity and prover complexity are unchanged.

Remark 1.7. *A PCP can be seen as an 1-round IOP where the verifier sends an empty message and the prover answers with a probabilistically checkable proof π. Since it is not per se a round of interaction, we may also view a PCP as a "half-round IOP". An r-round interactive proof (IP) can also be seen a r-round IOP, where the verifier queries each location of the prover's messages.*

1.2.5 Interactive oracle proofs of proximity

The notion of IOP of Proximity is the natural extension of the one of PCP of Proximity.

Definition 1.8 (IOP of Proximity (IOPP))**.** *An r-round IOP of Proximity (IOPP) system for a binary relation $\mathcal{R}$ with soundness error $\varepsilon : \mathbb{N} \to [0,1]$ and proximity parameter $\delta : \mathbb{N} \to [0,1]$ is a pair of probabilistic algorithms $(\mathcal{P}, \mathcal{V})$ such that:*

Completeness *For every $(x, w) \in \mathcal{R}$, $(\mathcal{P}(x, w), \mathcal{V}^w(x))$ is a r-round IOP with accepting probability 1.*

Soundness *For every (x, w) such that $\Delta(w, \mathcal{R}_{|x}) > \delta(n)$ and every unbounded prover $\widetilde{\mathcal{P}}$, $(\widetilde{\mathcal{P}}, \mathcal{V}^w(x))$ is a r-round IOP with accepting probability at most $\varepsilon(n)$, where n denotes the size of x.*

Compared to an IOP system (Definition 1.6), efficiency measures are defined similarly, except that queries to both the prover's messages and to the oracle w are taken into account.

IOP of Proximity for error-correcting codes. An IOP of Proximity for a family of codes $\mathcal{C}$ is an IOPP for a relation $\mathcal{R}$ which consists of pairs (x, w) where x is the description of a code C in $\mathcal{C}$, and w is a purported codeword of C. In particular, both the prover and the verifier receive as input the description of a code C (viewed as a subset of functions with some domain D and range Σ), and the prover tries to convince the verifier that some function $f : D \to \Sigma$, given as an oracle to the verifier, is a codeword of C.

Proximity oblivious IOPPs. Note that Definition 1.8 requires that the verifier rejects any witness w that is δ-far from $\mathcal{R}_{|x}$ with probability at least $1 - \varepsilon$. In particular, the verifier $\mathcal{V}$ implicitly receives the proximity parameter as auxiliary input, and its behavior is adapted accordingly (*e.g.* the query complexity may depend on δ). In most cases, an IOPP protocol can be decomposed into a basic protocol that does not use the proximity parameter, but is repeated several times[1]. As such, the proximity parameter only determines the number of times the basic test is repeated. Therefore, we also define "proximity oblivious" IOP of Proximity, where the proximity parameter is not given as input, but the soundness error is a function of the distance of w from $\mathcal{R}_{|x}$.

[1] Such a phenomenon more generally applies to *property testers*, namely sublinear-time probabilistic algorithms for deciding whether a given object (such as a graph) presents a given property or is far from the set of objects that have this property. In order to initiate a systematic study of property testers that can be decomposed into basic tests, Goldreich and Ron introduced the term *proximity oblivious testers* [GR11]. More details can be found in [Gol17, Section 1.3.3].

Definition 1.9 (IOPP, alternative definition)**.** *An* r*-round IOP of Proximity (IOPP) system for a binary relation* $\mathcal{R}$ *with soundness error* $\varepsilon : \mathbb{N} \times [0,1] \to [0,1]$ *is a pair of probabilistic algorithms* $(\mathscr{P}, \mathscr{V})$ *such that:*

Completeness *For every* $(\mathsf{x},\mathsf{w}) \in \mathcal{R}$, $(\mathscr{P}(\mathsf{x},\mathsf{w}), \mathscr{V}^{\mathsf{w}}(\mathsf{x}))$ *is a* r*-round IOP with accepting probability 1.*

Soundness *For every* (x,w) *such that* $\delta := \Delta(\mathsf{w}, \mathcal{R}_{|\mathsf{x}}) > 0$ *and every unbounded prover* $\widetilde{\mathscr{P}}$, $(\widetilde{\mathscr{P}}, \mathscr{V}^{\mathsf{w}}(\mathsf{x}))$ *is a* r*-round IOP with accepting probability at most* $\varepsilon(n,\delta)$*, where* n *denotes the size of* x*.*

1.2.6 Succinct non-interactive arguments from IOPs

From information-theoretic proof systems to cryptographic arguments. Probabilistically checkable proofs and interactive oracle proofs are information-theoretic proof systems, in the sense that no assumption is made on the computational resources of an adversarial prover. However, there are still assumptions that are made on the prover. For instance, for PCPs and IOPs, it is assumed that once a prover gives oracle access to some message to the verifier, the future answers to the verifier's queries are entirely determined and indeed correspond to the information that, in a way, the prover has committed to. This idealized behavior of the IOP prover is enforced in practice using some cryptographic primitives, in particular cryptographic commitments. By using cryptography, soundness holds against computationally bounded prover, which gives a so-called *argument system*. Informally, an argument system for a non-deterministic language $\mathcal{L}$ is *succinct* if its communication complexity is polylogarithmic in the sizes of the instance and the witness. By relying on PCP theorem [AS92, ALM+98], Kilian and Micali [Kil92, Mic95] showed the existence of non-interactive succinct arguments for **NP** in the random oracle model.

The BCS transformation. Ben-Sasson, Chiesa and Spooner [BCS16] showed that a public-coin IOP $(\mathscr{P}, \mathscr{V})$ can be transformed into a non-interactive argument, with unconditional security in the random oracle model. The *BCS transformation* [BCS16] is a generalization of the Fiat-Shamir paradigm [FS86] and of the constructions of Kilian [Kil92], Micali [Mic00] and Valiant [Val08]. We merely sketch the BCS transformation, and refers the reader to [BCS16] for details and formal proofs.

The BCS transformation can be thought in terms of two distinct components. The first step is to compile a public-coin IOP system into an *interactive argument*. Each oracle sent by the prover $\mathscr{P}$ is realized with a cryptographic commitment scheme that allows partial openings. Such commitments schemes are based on hash trees (also known as Merkle trees) and collision-resistant hash functions. This way, an interactive argument $(\mathscr{P}', \mathscr{V}')$ can be derived from an interactive oracle proof system as follows:

- Whenever the prover $\mathscr{P}$ sends an oracle message π_i to the verifier $\mathscr{V}$, the prover $\mathscr{P}'$ sends a commitment c_i (a hash digest representing the root of the Merkle tree) of the message π_i instead.
- After all oracles have been committed, the verifier $\mathscr{V}$ of the IOP system queries them at randomly chosen locations. When the $\mathscr{V}$ queries a given location j of a prover's message π_i, the prover $\mathscr{P}'$ opens the corresponding commitment c_i at the relevant location, by providing a Merkle path asserting that the value $\pi_i(j)$ was indeed committed under commitment c_i (a sequence of hash digests of length logarithmic in the length of π_i).

The second component consists in removing the interaction from $(\mathscr{P}', \mathscr{V}')$, in the spirit of the Fiat-Shamir paradigm [FS86] for interactive proof systems. The random messages of the verifier $\mathscr{V}'$ are simulated by the prover $\mathscr{P}'$ on its own: at each round, a random oracle is called on the partial transcript of the protocol generated up to this round, and the output of the random oracle replaces the corresponding verifier's random message.

Remark 1.10. *We recall that random oracles are used in formal security proofs requiring strong randomness assumptions on the output of the hash function as idealized replacements for cryptographic hash functions. While actual cryptographic hash functions are not random functions, random oracles are indeed instantiated by cryptographic hash functions in practice. Gentry and Wichs demonstrated in [GW11] that the security of succinct non-interactive arguments cannot be proved under standard simple cryptographic assumptions (so-called* falsifiable assumptions*) via black-box reductions.*

This finally leads to a non-interactive argument in the random oracle model. Let us briefly discuss communication complexity. Suppose that $(\mathscr{P}, \mathscr{V})$ is an IOP system with alphabet Σ, query complexity q and length l (counted in field elements). Assuming that a random oracle outputs κ bits, the size (in bits) of the succinct non-interactive argument derived from the IOP $(\mathscr{P}, \mathscr{V})$ using the BCS transformation is

$$O\left(\mathsf{q} \log |\Sigma| + \kappa \mathsf{q} \log \mathsf{l}\right).$$

The transformation from [BCS16] preserves knowledge soundness, meaning that an IOP of knowledge gives a non-interactive argument of knowledge in the random oracle model.

1.3 Error-correcting codes

We recall some elementary notions and notations related to error-correcting codes. More details can be found in [MS77].

1.3.1 Basic notions of coding theory

Codes. An error-correcting code (or simply a *code*) C is a non-empty set of functions $f : D \to \Sigma$, where D and Σ are finite sets referred to the *domain* and the *alphabet* of C, respectively. A function $f \in \Sigma^D$ which belongs to C is called a *codeword* of C. The *message length* of C is $k \coloneqq \log_{|\Sigma|} |C|$, its *block length* (or *length* for short) is $n \coloneqq |D|$, and its *(minimum) distance* is

$$d \coloneqq \min_{\substack{f, f' \in C \\ f \neq f'}} \left|\left\{x \in D \mid f(x) \neq f'(x)\right\}\right|.$$

The *rate* $\rho(C)$ of a code C is the proportion of non-redundant information in a codeword, namely $\rho(C) \coloneqq \frac{k}{n}$. The *relative distance* of C is $\Delta(C) \coloneqq \frac{d}{n}$.

Remark 1.11. *In coding theory, a code is often defined as a non-empty subset of Σ^n. The functional representation is particularly convenient for us, since applications presented in this manuscript rely on evaluation codes whose domain D is endowed with a specific algebraic structure.*

Linear codes. An important subclass of codes are linear codes. The alphabet of a linear code is a finite field, say $\mathbb{F}_q$. A code $C \subset \mathbb{F}_q^D$ is *linear* if C is an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^D$ (such a code is also called a q-ary code). Given a linear code $C \subset \mathbb{F}_q^D$, its message length is also its dimension as an $\mathbb{F}_q$-linear space, and its minimum distance is the minimum Hamming weight of its non-zero codewords. We say that a code C is a $[n,k,d]_q$-code if it is a linear code over $\mathbb{F}_q$ with length n, dimension k and minimum distance d.

Let C be an $\mathbb{F}_q$-linear code of block length n and dimension k. A *generator matrix* for C is a $k \times n$ matrix with coefficients in $\mathbb{F}_q$ whose rows form a basis of C. The dual code $C^\perp$ of C is the linear space composed by all elements that are orthogonal to the codewords of C. A *parity check matrix* for C is a generator matrix of its dual code $C^\perp$.

Let $\mathcal{L}$ be a vector space over $\mathbb{F}_q$ consisting of functions defined over some superset of a domain D, and taking values in $\mathbb{F}_q$. An evaluation map with respect to D is an $\mathbb{F}_q$-linear map

$$\begin{aligned} \mathrm{ev}_D \colon \mathcal{L} &\to \mathbb{F}_q^D \\ w &\mapsto w_{|D}, \end{aligned}$$

where $w_{|D} \colon D \to \mathbb{F}_q$ is the restriction of the function w to the domain D. In this thesis, we will only be interested in linear codes that are defined as the image of an *injective* evaluation map. Thus, for any codeword f of a code $C \coloneqq \mathrm{ev}_D(\mathcal{L})$, there will be a unique element of $w \in \mathcal{L}$ such that $\mathrm{ev}_D(w) = f$. Such element w is called the message associated to f.

Families of codes. A family of codes is an infinite collection $\mathcal{C} = \{C_i\}_{i \in \mathbb{N}}$ where C_i is a $[n_i, k_i, d_i]_{q_i}$-code with $n_{i+1} > n_i$. The rate $\rho(\mathcal{C})$ and the (relative) distance $\Delta(\mathcal{C})$ of an infinite family $\mathcal{C}$ are

$$\rho(\mathcal{C}) \coloneqq \inf_i \frac{k_i}{n_i} \qquad \text{and} \qquad \Delta(\mathcal{C}) \coloneqq \inf_i \frac{d_i}{n_i}.$$

We say that $\mathcal{C}$ has *constant rate* and *constant (relative) distance* if $\rho(\mathcal{C}) > 0$ and $\Delta(\mathcal{C}) > 0$. Code families *over a fixed alphabet* are family of codes where $q_i = q$ for all i and some fixed q. A family of codes with constant rate, constant relative distance and constant alphabet size is said to be *asymptotically good*.

Product codes. There are several ways of combining two linear codes to obtain a new one (see e.g. [MS77]). One of them consists in taking the tensor product of the parity check matrices of two linear codes, hence is known as the tensor product of two codes.

Definition 1.12 (Tensor product of codes)**.** *Given two linear codes $C_1 \subseteq \mathbb{F}_q^{n_1}$ and $C_2 \subseteq \mathbb{F}_q^{n_2}$, the tensor product code of C_1 and C_2, denoted $C_1 \otimes C_2$, consists of matrices M whose each row belongs to C_2 and each column belongs to C_1. Given an integer $m \geqslant 1$ and a code $C \subseteq \mathbb{F}_q^n$, we denote by $C^{\otimes m}$ the m-wise tensor product of C, where $C^{\otimes m}$ is inductively defined by*

$$C^{\otimes 1} \coloneqq C \qquad \text{and} \qquad C^{\otimes m} \coloneqq C^{\otimes m-1} \otimes C \quad \text{for } m > 1.$$

We will sometimes simply say "product codes" as a shorthand for "tensor product of codes". Parameters of such codes are well-known (details can be found in [Mei13]). Given a $[n_1, k_1, d_1]_q$-code C_1 and a $[n_2, k_2, d_2]_q$-code C_2, the code $C_1 \otimes C_2$ is a $[n_1 n_2, k_1 k_2, d_1 d_2]_q$-code. Similarly, if C is a $[n,k,d]_q$-code, then $C^{\otimes m}$ is a $[n^m, k^m, d^m]_q$-code.

Polynomial codes. A concrete approach to construct linear codes is to consider evaluations of bounded degree polynomial functions. Given a linear space of polynomials $\mathcal{L} \subset \mathbb{F}_q[X_1, \ldots, X_m]$ and a domain $D \subset \mathbb{F}_q^m$, let us consider the code $C \subset \mathbb{F}_q^D$ defined as the image of the evaluation map $\mathrm{ev}_D \colon \mathcal{L} \to \mathbb{F}_q^D$ satisfying, for all $P \in \mathcal{L}$,

$$\begin{aligned} \mathrm{ev}_D(P) \colon D &\to \mathbb{F}_q \\ \boldsymbol{x} &\mapsto P(\boldsymbol{x}). \end{aligned}$$

We call such a code C a *polynomial code*. Assuming that ev_D is an injective map, the dimension of the code $C = \mathrm{ev}_D(\mathcal{L})$ is the dimension of $\mathcal{L}$ as $\mathbb{F}_q$-vector space. Special families of polynomial codes are often given a name, as it the case for the well-known Reed-Solomon codes [RS60]. We will sometime write "RS code" as a shorthand for "Reed-Solomon code".

Definition 1.13 (Reed-Solomon code)**.** *Given $L \subseteq \mathbb{F}_q$ and $k \leqslant |L|$, we denote by $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ the Reed-Solomon (RS) code over alphabet $\mathbb{F}_q$ defined by*

$$\mathsf{RS}\left[\mathbb{F}_q, L, k\right] \coloneqq \left\{ f \in \mathbb{F}_q^L \mid \exists P \in \mathbb{F}_q[X]_{<k} \text{ s.t. } \forall x \in L,\, f(x) = P(x) \right\}.$$

The code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ is a $[n, k, d]_q$-code, where $n \coloneqq |L|$ and $d = n - k + 1$. The minimum distance of a Reed-Solomon code can be obtained by recalling that a univariate polynomial of degree less than k has at most $k - 1$ roots. Thus, the $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ has rate $\rho = \frac{k}{|L|}$ and relative minimum distance $\lambda = 1 - \frac{k-1}{|L|}$.

Next, we define two families of *multivariate polynomial codes*. The first one consists of evaluations of polynomial of degree bounded *in each variable*.

Definition 1.14 (Tensor product of Reed-Solomon code)**.** *Given $L_1, L_2, \ldots, L_m \subset \mathbb{F}_q$ and positive integers $m, k_1, k_2, \ldots, k_m$ such that $k_i \leqslant |L|$ for all $i \in [\![1, m]\!]$, the tensor product of Reed-Solomon codes*

$$\mathsf{RS}\left[\mathbb{F}_q, L_1, k_1\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L_2, k_2\right] \otimes \cdots \otimes \mathsf{RS}\left[\mathbb{F}_q, L_m, k_m\right]$$

is the linear space

$$\left\{ f \in \mathbb{F}_q^{L^m} \mid \exists P \in \mathbb{F}_q[\boldsymbol{X}],\, \deg_{X_i} P < k_i, i \in [\![1, m]\!] \text{ , such that } \forall \boldsymbol{x} \in L_1 \times L_2 \times \cdots \times L_m,\, f(\boldsymbol{x}) = P(\boldsymbol{x}) \right\}.$$

In particular, a m-wise tensor product code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$ has length $|L|^m$, dimension k^m, rate $\left(\frac{k}{|L|}\right)^m$ and relative distance $\left(1 - \frac{k-1}{|L|}\right)^m$.

Reed-Muller codes consist of evaluation of multivariate polynomials with coefficients in $\mathbb{F}_q$ of *bounded total degree*. The classical definition of (generalized) Reed-Muller codes involves evaluations over the whole finite field. We introduce here codes whose support is $L^m \subseteq \mathbb{F}_q^m$, where L may be much smaller than $\mathbb{F}_q$. This is an easy generalization, and we call these codes *short Reed-Muller codes*.

Definition 1.15 (Short Reed-Muller code)**.** *A* short Reed-Muller code *with support $L^m \subseteq \mathbb{F}_q^m$ is defined as follows*

$$\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right] \coloneqq \left\{ f \in \mathbb{F}_q^{L^m} \mid \exists P \in \mathbb{F}_q[\boldsymbol{X}],\, \deg P < k \text{ s.t. } \forall \boldsymbol{x} \in L^m,\, f(\boldsymbol{x}) = P(\boldsymbol{x}) \right\}.$$

If $k \leqslant |L|$, the evaluation map from the space of multivariate polynomials of total degree less than k to the space of functions $\mathbb{F}_q^{L^m}$ is injective, thus the dimension of $\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$ is $\binom{m+k-1}{m}$. A bound on the minimum distance of $\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$ follows from Lemma 1.19, which states that any non-zero multivariate polynomial $P \in \mathbb{F}_q[\boldsymbol{X}]$ of total degree less than q cannot vanish in more than $\frac{\deg P}{|L|}$ fraction of L^m. The code $\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$ has length $|L^m|$, rate $\binom{m+k-1}{m}|L|^{-m}$ and relative distance at least $1 - \frac{k-1}{|L|}$.

Remark 1.16. *The setting where the support $L^m \subset \mathbb{F}_q^m$ with $|L| \ll \left|\mathbb{F}_q\right|$ is not commonly encountered in coding theory. We introduce the non-standard term* short Reed-Muller codes *to emphasize this fact. Notice that, strictly speaking, short Reed-Muller codes correspond to* punctured *codes, and not* shortened *codes.*

Algebraic geometry codes. A part of the present study is dedicated to *algebraic geometry (AG) codes* [Gop77]. Algebraic geometry codes are a strict generalization of Reed-Solomon codes, but can be defined over much smaller (even constant-size) alphabets. They are notorious for achieving bounds that are better than those known from probabilistic constructions of codes [TVZ82]. Algebraic geometry codes correspond to evaluations of functions of a Riemann-Roch space on a suitable set of points on an algebraic variety. A formal definition of those codes require an introduction of several notions associated to algebraic geometry and the theory of algebraic function fields. Since we will not deal with AG codes before Chapter 5, we defer their definition to Section 5.2.

1.3.2 Locally testable codes

Locally testable codes (LTCs) are codes admitting a sublinear algorithm for the task of testing membership of the code. Informally, a LTC has a local tester (a probabilistic oracle machine) that makes a small number of queries to an oracle function that represents a purported codeword f of a code C, and rejects with high probability if f is far from any codeword of C. Constructions of locally testable codes are closely related to those of probabilistically checkable proofs (for more detail on this, we refer the reader to [Gol17, Chapter 13]).

We give a definition of a locally testable code from [Vid15] which is tailored for linear codes.

Definition 1.17 (Locally testable codes)**.** *Let $C \subset \mathbb{F}^D$ be a linear code, $\mathsf{q} \in \mathbb{N}$ and $\varepsilon \in (0,1]$. For a subset $Q \subset D$, we denote by*

$$C_{|Q} = \left\{f_{|Q} \mid f \in C\right\}.$$

Let $\mathscr{V}_C$ be a probabilistic (non-adaptive) polynomial time machine which is given oracle access to a purported codeword $f \in \Sigma^D$. Suppose that $\mathscr{V}_C$ generates a query set $Q \subseteq D$ of size q and query f on Q (the restriction of f to Q is called the local view *of $\mathscr{V}_C$). Then, $\mathscr{V}_C$ accepts if and only if $f_{|Q} \in C_{|Q}$. For some integer $\mathsf{q} \leqslant |D|$ and $\varepsilon \in (0,1]$, we say that $\mathscr{V}_C$ is a $(\mathsf{q}, \varepsilon)$-local tester for C if the two following conditions are satisfied.*

Completeness: *For every $f \in C$, $\mathscr{V}_C^f$ accepts with probability 1.*

Soundness: *For any function $f \notin C$, $\mathscr{V}_C^f$ accepts with probability at most $1 - \varepsilon \cdot \Delta(f, D)$.*

A q-query locally testable code (LTC) is a code C which admits a $(\mathsf{q}, \varepsilon)$-local tester for some $\varepsilon \in (0,1]$.

While Definition 1.17 allows tester that are not "local" in the sense that q can be as large as the block length $n := |D|$, the $(\mathsf{q}, \varepsilon)$-local testers that are of interest are obviously those whose query complexity and running time are sublinear in n.

The local testability of a code can be strengthened by the notion of *robustness*. Loosely speaking, robustness means that, for any word that is far from the code, the local view of a local tester is expected to be far from the set of accepting local views.

Definition 1.18 (Robustness)**.** *We say that a* q*-query locally testable code* $C \subset \mathbb{F}^D$ *is* α*-robust for some* $\alpha \in [0,1]$ *if, for any function* $f \notin C$*, we have*

$$\mathbf{E}\left[\Delta(f_{|Q}, C_{|Q})\right] \geqslant \alpha \cdot \Delta(f, C),$$

where the expectation is taken over the collection of subsets $Q \subseteq D$ *of size* q*.*

1.4 Some preliminary technical lemmas

1.4.1 Zeroes of polynomials over finite fields

Polynomial identity lemma. The following lemma is commonly referred to as the Schwartz-Zippel lemma. It is also known as DeMillo-Lipton-Schwartz–Zippel lemma[2] [DL78, Zip79, Sch80]. A standard proof is by induction on the number of variables. Alternatively, a direct proof was given by Moshkovitz [Mos10].

Lemma 1.19. *Let* $\mathbb{F}_q$ *be a finite field,* $S \subset \mathbb{F}_q$*, and* $P \in \mathbb{F}_q[\boldsymbol{X}]$ *a non-zero polynomial of (total) degree at most* d*. Then*

$$\Pr_{z \in S^m}[P(z) = 0] \leqslant \frac{d}{|S|}.$$

We give two immediate consequences of Lemma 1.19.

Corollary 1.20. *If* $P \in \mathbb{F}_q[\boldsymbol{X}]$ *has degree at most* d *and*

$$\Pr_{z \in S^m}[P(z) = 0] > \frac{d}{|S|},$$

then P *is the zero polynomial. Moreover, if* $P, Q \in \mathbb{F}_q[\boldsymbol{X}]$ *are polynomial of degree at most* d *and*

$$\Pr_{z \in S^m}[P(z) = Q(z)] > \frac{d}{|S|},$$

then the polynomials P *and* Q *are equal. Namely, two distinct polynomials of degree at most* d *agree on at most* $\frac{d}{|S|}$*-fraction of points in* S^m*.*

[2] As pointed out in [AJMR19, Section 3.1], assigning the right credits for this basic yet fundamental result is a somewhat tricky task.

Polynomials vanishing on cartesian products. We now give a technical lemma about multivariate polynomials vanishing on product sets. It is a key ingredient of Alon's combinatorial Nullstellensatz [Alo99].

Lemma 1.21. *Let $P \in \mathbb{F}_q[\boldsymbol{X}]$ be a polynomial of degree d in each variable. Let $H_1, \dots, H_m \subset \mathbb{F}_q$ and, for each $i \in [\![1, m]\!]$, consider $Z_i \in \mathbb{F}_q[X]$ the unique monic polynomial of degree $|H_i|$ that vanishes on H_i, i.e.*

$$Z_i(X) \coloneqq \prod_{h \in H_i} (X - h).$$

Then, the two following statements are equivalent.

i) For all $\boldsymbol{x} \in H_1 \times H_2 \times \cdots \times H_m$, $P(\boldsymbol{x}) = 0$;
ii) There exist m polynomials $Q_1, \dots, Q_m \in \mathbb{F}_q[\boldsymbol{X}]$ of individual degree at most d such that

$$P(\boldsymbol{X}) = \sum_{i=1}^{m} Q_i(\boldsymbol{X}) Z_i(X_i).$$

Proof. See [BS08, Lemma 4.11]. □

The result also holds with tighter degree bounds on the polynomials $Q_1, \dots, Q_m$, but the stated bounds will suffice for our purposes.

1.4.2 Distance preservation and random linear combinations

We state some preliminaries results about distance of random linear combination to linear subspaces. These results have applications to efficient constructions of proof systems, and in particular to interactive proximity tests. In such a context, a prover knows a tuple of functions $(u_1, \dots, u_l) \in \left(\mathbb{F}^D\right)^l$ and claims that the functions $u_1, \dots, u_l$ are all closed to a given code $V \subset \mathbb{F}^D$. On the other side, the verifier can query those functions at some desired locations. Instead of running a proximity test on each function, the verifier randomly samples $r_1, \dots, r_l \in \mathbb{F}$ and the prover and the verifier run a proximity test to check whether $u \coloneqq \sum r_i u_i$ is itself close to the code V.

Lemma 1.22 ([RVW13, AHIV17])**.** *Let $\delta \in [0, 1]$, $u_1, \dots, u_l \in \mathbb{F}_q^n$ and $V \subset \mathbb{F}_q^n$ be a linear subspace. If $\Delta(u_j, V) > \delta$ for some $j \in [\![1, l]\!]$, then*

$$\Pr_{z_1, \dots, z_l \in \mathbb{F}_q} \left[\Delta\left(\sum_{i=1}^{l} z_i u_i, V \right) < \frac{\delta}{2} \right] \leqslant \frac{1}{q}.$$

Proof. Assume $\Delta(u_j, V) > \delta$. For $z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_l \in \mathbb{F}_q$, we set $y \coloneqq \sum_{\substack{i=0 \\ i \neq j}}^{l} z_i u_i$. By way of contradiction, assume there are two distinct elements $z_j, z_j' \in \mathbb{F}_q$ such that $\Delta(z_j u_j + y, V), \Delta(z_j' u_j + y, V) < \delta/2$. Then, by the triangle inequality, $\Delta((z_j - z_j') u_j, V) \leqslant \delta/2 + \delta/2$. This contradicts the fact that $\Delta(u_j, V) > \delta$. Thus, conditioned on any choice of y, there is at most one z_j such that $\Delta(z_j u_j + y, V) < \frac{\delta}{2}$. □

Lemma 1.22 holds for *any* values δ, but incurs a factor 2 loss in the proximity parameter δ. As a corollary of the following result, one can prove that the same kind of statement holds with only a small additive loss in the proximity parameter, but bounded values of δ.

Lemma 1.23 ([BGKS20, Lemma 2]). *Let $C \subset \mathbb{F}_q^D$ be a linear code of distance $\lambda = \Delta(C)$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < 1/3$ and $\delta < 1 - (1 - \lambda + \varepsilon)^{1/3}$. For any functions $u_0, u_1 \in \mathbb{F}_q^D$ satisfying*

$$\Pr_{z \in \mathbb{F}_q} \left[\Delta(u_0 + z u_1, C) < \delta - \varepsilon \right] \geqslant \frac{2}{\varepsilon^2 q}, \tag{1.1}$$

there exist $T \subset D$ and $v_0, v_1 \in C$, such that

- $|T| \geqslant (1 - \delta)\,|D|$,
- *for each $i \in \{0, 1\}$, $u_{i|T} = v_{i|T}$.*

Consequently, we have $\Delta(u_0, C), \Delta(u_1, C) \leqslant \delta$ and, for all $z \in \mathbb{F}_q$, $\Delta(u_0 + z u_1, v_0 + z v_1) \leqslant \delta$.

In words, Lemma 1.23 states that if there are enough values of z such that the linear combination $u_0 + z u_1$ of two functions $u_0, u_1 \in \mathbb{F}_q^D$ is close to a code C, then there exists a large subset of coordinates $T \subset D$ such that $u_{0|T}$ and $u_{1|T}$ are codewords of the punctured code

$$C_{|T} \coloneqq \left\{ f_{|T} : T \to \mathbb{F}; f \in C \right\}.$$

Corollary 1.24. *Let C, ε, δ satify the assumptions of Lemma 1.23, and let $u_1, \ldots, u_l \in \mathbb{F}_q^n$. If there exists $j \in [\![1, l]\!]$ such that $\Delta(u_j, C) > \delta$, then*

$$\Pr_{z_1, \ldots, z_l \in \mathbb{F}_q} \left[\Delta \left(\sum_{i=1}^{l} z_i u_i, C \right) < \delta - \varepsilon \right] < \frac{2}{\varepsilon^2 q}.$$

Proof. Assume $\Delta(u_j, V) > \delta$. Fix $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_l \in \mathbb{F}_q$ and set $u \coloneqq \sum_{\substack{i=0 \\ i \neq j}}^{l} z_i u_i$. By the contrapositive of Lemma 1.23, we have

$$\left| \left\{ z_j \in \mathbb{F}_q; \Delta\left(u + z_j u_j, C\right) < \delta \right\} \right| < 2/\varepsilon^2.$$

We deduce that

$$\left| \left\{ (z_1, \ldots, z_l) \in \mathbb{F}_q^l; \Delta \left(\sum_{i=1}^{l} z_i u_i, C \right) < \delta \right\} \right| < \frac{2}{\varepsilon^2} q^{l-1}.$$

□

CHAPTER 2

Reed-Solomon proximity testing and application to computational integrity

Given a code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$, the Reed-Solomon proximity problem consists in (probabilistically) distinguishing between the cases where a purported codeword f belongs to $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ and f is far from $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$. A PCPP (resp. IOPP) for Reed-Solomon codes is a PCPP (resp. IOPP) for the relation

$$\mathcal{R}_{\mathsf{RS}} \coloneqq \left\{\left((\mathbb{F}_q, L, k), f\right) \mid L \subseteq \mathbb{F}_q, k \in \mathbb{N} \setminus \{0\}, k < |L|, f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]\right\}.$$

(See Definition 1.5 and Definition 1.9 for definitions of PCP of Proximity and IOP of Proximity).

In the first section of this chapter, we discuss solutions to the Reed-Solomon proximity problem. We briefly discuss existing solutions in terms of local testers and PCP of Proximity, then describe in more details the IOP of Proximity for Reed-Solomon codes proposed by Ben-Sasson, Bentov, Horesh and Riabzev [BBHR18a]. In the second section, we illustrate the role of the Reed-Solomon proximity problem in verifiable computing, by providing a simplified exposition of an IOP with logarithmic verification from [BBHR19], which relies on the IOPP for Reed-Solomon codes of [BBHR18a].

2.1 Reed-Solomon proximity testing in the IOP model

2.1.1 Some background on Reed-Solomon proximity testing

A simple and "optimal" test. Let us first discuss which query complexity can be expected when a verifier has only oracle access to a purported codeword f of $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$, without any additional information. Then testing proximity to the code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ requires the verifier to read at least $k+1$ entries of f. Indeed, there is always a polynomial of degree less than k which agrees with k values of f. The following simple test shows that $k+1$ queries are not only necessary, but also sufficient.

Construction 2.1 (Direct test)**.**
Input: *An oracle function $f \in \mathbb{F}^L$, expected to belong to a RS code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$.*

1. *Let $I \subset L$ be an arbitrary set of k elements. Compute the polynomial $P(X)$ of degree less than k interpolating the set of points $\{(x, f(x))\}_{x \in I}$;*
2. *Sample uniformly at random $\alpha \xleftarrow{\$} L$ and query f for the value $f(\alpha)$;*
3. *Output accept if and only if $P(\alpha) = f(\alpha)$.*

Lemma 2.2. *If f is a codeword of $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$, the test presented in Construction 2.1 always accepts. If f is δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$, the test accepts with probability at most $1-\delta$.*

Proof. It is easy to see that the test always accepts when $f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]$. Now assume that $\Delta(f, \mathsf{RS}\left[\mathbb{F}_q, L, k\right]) > \delta$. Then the probability over r that $P(r) \neq f(r)$ is equal to $\Delta(P_{|L}, f)$. We conclude the proof by noticing that the distance between $P_{|L} \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ and f is itself greater than δ. □

Remark 2.3. *Note that there are some values of $\delta \in [0,1]$ for which it is not relevant to ask whether a function f is δ-close to a given Reed-Solomon code. Indeed, it can be deduced from Lagrange interpolation that any function $f \in \mathbb{F}_q^L$ is at distance at most $1-\rho$ of a Reed-Solomon code of rate ρ. The maximum error distance*

$$\max\left\{\Delta(f, C) \mid f \in \mathbb{F}_q^L\right\}$$

is called the (relative) covering radius *of the code.*

The test presented in Construction 2.1 is optimal in the sense that it makes as few queries as necessary. However, recall that the interesting regime for applications to proof systems is when $k = \Theta(|L|)$. In that case, a $\Theta(k)$-query test is not satisfactory.

Reed-Solomon proximity testing in the PCPP model. In 2008, Ben-Sasson and Sudan constructed a PCP of Proximity for Reed-Solomon codes [BS08] with polylogarithmic query complexity, based on the bivariate low-degree test of Polishchuk and Spielman [PS94]. Several works have built on those of [BS08], resulting in PCPPs for Reed-Solomon codes with quasilinear-size proofs and constant query complexity [BS08, Din07], quasilinear time prover [BCGT13] and polylogarithmic verifier [BGH+04, Mie09].

Informally, the PCPP of [BS08] uses a technique called *proof composition* ([AS92, BGH+04, DR04]) to reduce a given proximity testing problem to a similar but significantly smaller problem. We briefly describe the PCPP of Ben-Sasson and Sudan. The idea is to test whether a function $f : L \to \mathbb{F}_q$ is close to a Reed-Solomon code of dimension k by testing whether a function g is close to a tensor product of two Reed-Solomon codes of dimension $k' \approx \sqrt{k}$, *i.e.* the space of evaluations of *bivariate* polynomials of degree less than k' in each variable. The following lemma shows how to define a bivariate polynomial which "captures the information" of a univariate one.

Lemma 2.4 ([BS08, Proposition 6.3])**.** *Given any pair of polynomials $P(X), q(X)$, there exists a unique bivariate polynomial $Q(X,Y)$ with $\deg_X Q < \deg P$ and $\deg_Y Q \leqslant \lfloor \deg P / \deg q \rfloor$ such that $P(X) = Q(X, q(X))$.*

Typically, the polynomial $q(X)$ in Lemma 2.4 is a well-chosen polynomial of degree $\deg q \approx \sqrt{k}$. Given a bivariate function g evaluated over a product set $D_1 \times D_2, D_1, D_2 \subset \mathbb{F}_q$, a bivariate low-degree test is (for the sake of this informal discussion) a randomized procedure testing the proximity of g to a product code $\mathsf{RS}\left[\mathbb{F}_q, D_1, k_1\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, D_2, k_2\right]$, where $k_1, k_2 \approx \sqrt{k}$. A bivariate low-degree test was proposed by Polishchuk and Spielman [PS94]. It consists in checking whether the restriction of the bivariate function g to a randomly sampled row or column is itself close to the evaluation of a univariate polynomial of degree less than k'. The soundness of the bivariate test of [PS94] relies on the following result.

Lemma 2.5 ([PS94])**.** *There exists a universal constant $c_0 \geqslant 1$ such that the following holds. Given $D_1, D_2 \subset \mathbb{F}_q$ and integers $k_1 < \frac{|D_1|}{4}, k_2 < \frac{|D_2|}{8}$, consider*

$$C_1 \coloneqq \mathsf{RS}\left[\mathbb{F}_q, D_1, k_1\right] \text{ and } C_2 \coloneqq \mathsf{RS}\left[\mathbb{F}_q, D_2, k_2\right].$$

For every function $g : D_1 \times D_2 \to \mathbb{F}_q$*, we have*

$$\Delta(g, C_1 \otimes C_2) \leqslant c_0 \left[\Delta\left(g, C_1 \otimes \mathbb{F}_q^{D_2}\right) + \Delta\left(g, \mathbb{F}_q^{D_1} \otimes C_2\right)\right].$$

To summarize, a univariate low-degree test for degree k can be reduced to a bivariate low-degree test with degree bound $k' \approx \sqrt{k}$ in each variable, and such bivariate test is itself a reduction to a univariate low-degree test, but with degree bound k'. By recursing $\Theta(\log\log k)$ times (using proof composition for PCPPs [AS92, BGH+04, DR04]), the initial proximity problem for a Reed-Solomon code of dimension k is eventually reduced to a problem for a Reed-Solomon code of constant dimension.

The resulting PCP of Proximity for Reed-Solomon codes [BS08] of length n has quasilinear proof length and polylogarithmic query complexity. In the rest of the section, we will see how a verifier can test proximity to a Reed-Solomon code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ with $O(\log k)$ query complexity by interacting with a prover. Thus, this is an exponential improvement over the situation where no additional information is available to the verifier.

Remark 2.6. *A limitation of [BS08] is that each reduction consisting in reducing the size of the RS proximity testing problem causes a multiplicative factor loss to the distance parameter* δ*. Due to this multiplicative loss, at most* $\Theta(\log\log n)$ *recursive reductions could be applied (this underlying the choice of setting* $k' \approx \sqrt{k}$ *in the foregoing discussion). In [BCG+17], a natural IOPP version of the PCPP for Reed-Solomon codes of [BS08, Din07] is proposed, leading to* $O(n)$*-size proofs, same soundness as [BS08] and constant query complexity. Prover complexity is* $\Theta(n \cdot \mathrm{polylog}(n))$ *due to the aforementioned limitation on the number of recursions.*

In contrast, the IOPP for Reed-Solomon code constructed in [BBHR18a] has a strictly logarithmic number of rounds and a strictly linear-time prover. This is partially explained from the fact that [BBHR18a] managed to maintain the distance parameter over the rounds of interaction, in the sense that in [BBHR18a], the distance parameter is only impacted by an additive *term. In [BBHR18a, Section 2.2], the authors explained that the multiplicative soundness loss appearing in [BS08] is avoided by unbanlancing the degree in each variable of the bivariate polynomial obtained from Lemma 2.4, e.g. by setting* $\deg q = 2$ *instead of* $\deg q \approx \sqrt{k}$ *in the aforementioned lemma.*

2.1.2 Outline of the FRI protocol

Let R be a positive integer and $\rho \in (0,1)$ such that $\rho = 2^{-R}$. The FRI protocol is an IOPP for Reed-Solomon codes $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0 = \rho\,|L_0|\right]$ where L_0 is a coset of a subgroup in $(\mathbb{F}_q^\times, \times)$ or $(\mathbb{F}_q, +)$ of size a power of 2. The name FRI stands for "Fast Reed-Solomon IOPP" and recalls its similarity to the Fast Fourier Transform (FFT).

The FRI protocol operates over $O(\log(k_0))$ rounds, where each round reduces the problem of proximity to a code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ to the one of testing proximity to a code $\mathsf{RS}\left[\mathbb{F}_q, L', \frac{k}{2}\right]$ evaluated over domain of half the size, i.e. $|L'| = \frac{|L|}{2}$. For the protocol to work, we assume that k is divisible by 2. We give some intuition on the design of the FRI protocol for the case where $\mathbb{F}_q$ is a prime field of odd characteristic.

The ideas of the FRI protocol in a nutshell. First, the FRI protocol relies on a decomposition of univariate polynomials. Given $f \in \mathbb{F}_q^L$, consider $\widehat{f} \in \mathbb{F}_q[X]$ its interpolant polynomial (i.e. $\widehat{f}(x) =$

$f(x)$ for all $x \in L$ and $\deg \widehat{f} < |L|$). There exist $\widehat{g_0}, \widehat{g_1} \in \mathbb{F}_q[Y]$ such that

$$\deg \widehat{g_0} \leqslant \left\lfloor \frac{\deg \widehat{f}}{2} \right\rfloor, \quad \deg \widehat{g_1} \leqslant \left\lfloor \frac{\deg \widehat{f} - 1}{2} \right\rfloor$$

and

$$\widehat{f}(x) = \widehat{g_0}\left(x^2\right) + x \cdot \widehat{g_1}\left(x^2\right). \tag{2.1}$$

This well-known decomposition into even and odd coefficients (respectively, $\widehat{g_0}$ and $\widehat{g_1}$) is also used in the 2-radix FFT algorithm. Let us denote by g_0 and g_1 the evaluations of $\widehat{g_0}(Y)$ and $\widehat{g_1}(Y)$ on L', respectively.

Second, the evaluation domains are chosen to be compatible with this decomposition. Thus, consider L a union of cosets of the subgroup $\{1, -1\} \subset \mathbb{F}_q^\times$ (in particular, L is stable under negation) and set $L' \coloneqq \left\{x^2 \mid x \in L\right\}$. Since k is an even integer, in the case where $f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]$, we thus have $g_0, g_1 \in \mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$.

Finally, a proximity test for $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ can be reduced to a single proximity test for $\mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$ by forming a linear combination of g_0 and g_1. Specifically, define for any $z \in \mathbb{F}_q$ a *folding operator* $\mathbf{Fold}\left[\cdot, z\right]$ as follows:

$$\begin{aligned} \mathbf{Fold}\left[\cdot, z\right] : \mathbb{F}_q^L &\to \mathbb{F}_q^{L'} \\ f &\mapsto \mathbf{Fold}\left[f, z\right] \coloneqq g_0 + z g_1. \end{aligned}$$

The value $z \in \mathbb{F}_q$ will be randomly chosen by the verifier during the interactive phase of the IOPP. In the completeness case, and by linearity of the code $\mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$, the folded function $\mathbf{Fold}\left[f, z\right]$ of $f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ with respect to z is a codeword $\mathbf{Fold}\left[f, z\right] \in \mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$.

The key properties of the folding operators. Taking a random linear combination of g_0 and g_1 not only enables to reduce a low-degree test to a *single* low-degree test with smaller degree bound, but also ensures that the relative distance of $\mathbf{Fold}\left[f, z\right]$ to the code $\mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$ is *roughly the same* as the distance of f to $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$. To prove such a distance-preserving statement, one can rely on Lemma 1.23.

Another crucial property of the folding operator is that, for any $f \in \mathbb{F}^L$ and $y \in L'$, the values $g_0(y)$ and $g_1(y)$ can be computed from two values of f. To see this, let us consider the two *distinct* square roots $x, -x \in L$ of y. Then $g_0(y)$ and $g_1(y)$ can be computed by solving the system of two independent equations

$$\begin{cases} f(x) = g_0(y) + x g_1(y) \\ f(-x) = g_0(y) - x g_1(y) \end{cases}.$$

In summary, the folding operators are defined such that they satisfy the following three properties:

i) *Completeness:* For any random challenge $z \in \mathbb{F}_q$, the folding operator $\mathbf{Fold}\left[\cdot, z\right]$ maps codewords of $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ onto codewords of $\mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]$.
ii) *Local computability:* For any function $f \in \mathbb{F}_q^L, z \in \mathbb{F}_q$ and $y \in L'$, the evaluation of $\mathbf{Fold}\left[f, z\right]$ at y can be computed by making exactly 2 queries to f.

iii) *Distance preservation:* Except with small probability over z, if

$$\Delta(f, \mathsf{RS}\left[\mathbb{F}_q, L, k\right]) \geqslant \delta,$$

then

$$\Delta\left(\mathbf{Fold}\left[f, z\right], \mathsf{RS}\left[\mathbb{F}_q, L', k/2\right]\right) \geqslant (1 - o(1))\delta.$$

Remark 2.7. *These three properties were first formulated in [BKS18] to describe the FRI protocol, but the term "algebraic hash function" was used in place of "folding operator". Our preference for the term "folding operator" is motivated by the fact that the definition of the folding operator does not need to be algebraic*[1]*. In Chapter 3, we will show that one can construct an IOPP for a linear code whenever the code is endowed with folding operators with the three desired properties.*

Reduction to a constant-size problem from recursion. The protocol then goes as follows: the verifier sends a random challenge $z \in \mathbb{F}_q$ and the prover answers with an oracle function $f' : L' \to \mathbb{F}_q$, which is expected to be equal to $\mathbf{Fold}\left[f, z\right] : L' \to \mathbb{F}_q$. The possibility of determining any value of $\mathbf{Fold}\left[f, z\right]$ at a point $y \in L'$ with exactly two values of f suggests a natural consistency test at each round: the consistency between f and f' can be checked at a random location with only two queries to f and one query to f'.

At the next round, f' becomes the function to be "folded", and the process is repeated for r rounds. Each round halves the size of the proximity problem, eventually leading to a function f_r evaluated over a constant-size evaluation domain. This recursive process induces a sequence of Reed-Solomon codes of strictly decreasing length. The code rate remains unchanged. The final test consists in testing that f_r belongs to the last RS code.

The structure of the evaluation domain of the first code $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]$ must be chosen to allow iterated reductions over a logarithmic number of rounds. Therefore, L_0 is defined as a subgroup of $\mathbb{F}_q$ whose order is divisible by a large power of 2. Besides, k_0 is also assumed to be a large power of 2, so that halving the degree bound at each round always gives an even integer.

We provide some technical lemmas and definitions to define folding operators for Reed-Solomon codes, then formally describe the FRI protocol in Section 2.1.5.

2.1.3 Decomposition of univariate polynomials

Lemma 2.8 (Univariate decomposition)**.** *Let R be an integral domain, and let $q \in R[X]$ be a monic polynomial of degree l. For every $f \in R[X]$, the following two statements hold.*

1. *There exists a unique sequence of polynomials $(f_i(X))_{0 \leqslant i \leqslant \left\lfloor \frac{\deg f}{l} \right\rfloor}$ of degrees less than l such that*

$$f(X) = \sum_{i=0}^{\lfloor \deg f / l \rfloor} f_i(X) q(X)^i.$$

2. *There exists a unique sequence of polynomials $(g_j(X))_{0 \leqslant j < l}$ of degrees at most $\lfloor \deg f / l \rfloor$ such that*

$$f(X) = \sum_{j=0}^{l-1} X^j g_j(q(X)).$$

[1] For instance, [BCG20] proposed a proximity test based on a folding operation for tensor product of codes, where the only assumption on the base code is its $\mathbb{F}_q$-linearity. A notation similar to ours is used there.

Example 2.9. When $q(X) = X^2$, we get the decomposition of a polynomial $f \in \mathbb{F}_q[X]$ into two polynomials $g_0, g_1 \in \mathbb{F}_q[X]$ which correspond to the even and odd coefficients of f, respectively.

Proof of Lemma 2.8. We begin with the proof of the first item. As in [BS08, Proposition 6.3], we consider the Euclidean division of $f(X)$ by $(Y - q(X))$ in the polynomial ring $R[Y][X]$, i.e. with respect to the X variable. Polynomial division by a monic polynomial over an integral domain shares the same properties as polynomial division over a field. There exists a unique pair of polynomials $A,\ B \in R[X][Y]$ such that

$$f(X) = (Y - q(X))A(X,Y) + B(X,Y),$$

$\deg_X B < \deg q$ and $\deg_Y B \leqslant \lfloor \deg f/l \rfloor$. Writing $B(X,Y) = \sum_{i=0}^{\lfloor \deg f/l \rfloor} f_i(X)Y^i$ with $\deg f_i < \deg q$, and evaluating the above identity at $Y = q(X)$, gives the required decomposition. Its uniqueness follows from the one of the remainder B in the Euclidean division, as any other decomposition $\sum_{i=0}^{\lfloor \deg f/l \rfloor} f'_i(X)q(X)^i$ with the same degree bounds would induce a different remainder $\sum_{i=0}^{\lfloor \deg f/l \rfloor} f'_i(X)Y^i \neq B$. This concludes the proof of the first item.

The second item is proved using the decomposition of the first item. For $i \in [\![0, \lfloor \deg f/l \rfloor]\!]$, we can write $f_i(X) \coloneqq \sum_{j=0}^{l-1} a_{i,j}X^j$. Now, for $j \in [\![0, l-1]\!]$, define $g_j(X) \coloneqq \sum_{i=0}^{\lfloor \deg f/l \rfloor} a_{i,j}X^i$. We get

$$f(X) = \sum_{i=0}^{\lfloor \deg f/l \rfloor} \left(\sum_{j=0}^{l-1} a_{i,j}X^j \right) q(X)^i = \sum_{j=0}^{l-1} X^j \left(\sum_{i=0}^{\lfloor \deg f/l \rfloor} a_{i,j}q(X)^i \right) = \sum_{j=0}^{l-1} X^j g_j(q(X)).$$

The uniqueness of the sequence of polynomials $(g_j)_j$ follows from the uniqueness of $(f_i)_i$. □

2.1.4 Algebraic setting for polynomial codes

In this section, we provide common definitions and notations for two different settings allowing proof of proximity for polynomial codes. These two settings were introduced for the PCPP for Reed-Solomon codes [BS08], are used in the IOPP protocol of [BBHR18a] and will also play a role in our IOPPs for multivariate polynomial codes in Chapter 4.

Case 2.1 (Multiplicative subgroups)**.**
In this case, we assume that $\mathbb{F}_q$ has odd characteristic and admits a large multiplicative subgroup L_0 of order 2^n. In particular, prime fields $\mathbb{F}_p$ such that $p-1$ is divisible by a large power of 2 are abundant (see [BS08, Section 7.7]).

For any integer r, we define a sequence of evaluation domains $(L_i)_{0 \leqslant i \leqslant r}$ as: $L_{i+1} \coloneqq q_i(L_i)$ where $q_i(X) = X^2$. Let $A_i \subset L_i$ be a multiplicative subgroup of L_i of order $\deg q_i$, i.e. $A_i \coloneqq \{x \mid q(x) = 1\}$. Then each multiplicative coset of A_i is mapped to a single element of L_{i+1} by the map $x \mapsto q_i(x)$. We have that $|L_{i+1}| = \frac{1}{2}|L_i| = \frac{1}{2^i}|L_0|$.

Case 2.2 (Additive subgroups)**.**
In this case, we assume $\mathbb{F}_q$ has characteristic two admitting an additive subgroup of size 2^n for some positive integer n.

For any integer r, we define a sequence of evaluation domains $(L_i)_{0 \leqslant i \leqslant r}$ recursively as follows. For $i = 0$, we consider $L_0 \subset \mathbb{F}_q$ an additive coset of a subgroup of order 2^n of $(\mathbb{F}_q, +)$ and $A_0 \subset L_0$

a coset of a $\mathbb{F}_2$-affine subspace of dimension 1. Then, for $i \in [\![1, r]\!]$, we define $L_i := q_{i-1}(L_{i-1})$, where $q_{i-1}(X) := \prod_{a \in A_{i-1}} (X - a)$ and $A_{i-1} \subset L_{i-1}$ is a coset of a $\mathbb{F}_2$-affine subspace of dimension $\dim A_{i-1} = 1$.

Note that $q_i(X)$ is a so-called *affine subspace polynomial*, and in particular a linearized polynomial (see e.g., [LN97, Chapter 3.4] for details and properties). It has the form $X^2 + \alpha X + \beta$ for $\alpha, \beta \in \mathbb{F}_q$, and each additive coset of A_i is mapped to a single element of L_{i+1} by the map $x \mapsto q_i(x)$. Moreover, $\dim L_{i+1} = \dim L_i - \dim A_i = \dim L_i - 1$. As for the Case 2.2, we that $|L_{i+1}| = \frac{1}{2}|L_i| = \frac{1}{2^i}|L_0|$.

Remark 2.10. *In [BBHR18a], the protocol is parameterized by a so-called* localization parameter η. *We focus on the case $\eta = 1$ in this manuscript. In that paper, the degree of the maps q_i and the size of the set A_i defined above are set to be equal to 2^η. Moreover, instead of taking L_0 of size 2^n as we do, it is assumed that $|L_0| = 2^{2^\eta}$. Accordingly, the size of the sets L_{i+1} is the one of L_i divided by 2^η. The main properties of the FRI protocol in [BBHR18a, Theorem 1.3] are given for $\eta = 2$.*

The construction of [BBHR18a] can be generalized to subgroup L_0 of order c^n (for some constant c) by defining $q_i \in \mathbb{F}_q[X]$ of degree $\deg q_i = c$ (see [BBHR18a, Remark 1.4]).

2.1.5 The FRI protocol: description and analysis

We have now provided sufficient groundwork for implementing the ideas described in Section 2.1.2.

Let $k_0 = 2^r$ for some integer r, and define $k_{i+1} = k_i/2$ for $i \in [\![0, r-1]\!]$. In the rest of the section, we assume that we have a sequence $(L_i)_{0 \leqslant i \leqslant r}$ defined as per Section 2.1.4, depending on whether we are in Case 2.1 or Case 2.2.

Folding operators. We start by defining folding operators for Reed-Solomon codes and present their properties. Fix some $i \in [\![0, r-1]\!]$ and consider q_i as defined in Section 2.1.4.

Definition 2.11. *Let $L_i \subset \mathbb{F}_q$ and $q_i \in \mathbb{F}_q[X]$ as above. Let $f : L_i \to \mathbb{F}_q$ be an arbitrary function. Denote $\widehat{f}$ the interpolating polynomial of $f : L_i \to \mathbb{F}_q$. Let g_0 and g_1 be the evaluations on L_{i+1} of the polynomials $\widehat{g_0}$ and $\widehat{g_1}$ obtained by applying Lemma 2.8 to $\widehat{f} \in \mathbb{F}_q[X]$.*

For any $z \in \mathbb{F}_q$, we define the folding *of f as the function* **Fold** $[f, z] : L_{i+1} \to \mathbb{F}_q$ *such that for all $y \in L_{i+1}$,*

$$\textbf{Fold}\,[f, z]\,(y) = g_0(y) + z g_1(y).$$

Lemma 2.12. *Using notations of Definition 2.11 and Section 2.1.4, the folding operators defined in Definition 2.11 satisfy the following properties.*

1. *If $f \in \mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$, then for any $z \in \mathbb{F}_q$, we have* **Fold** $[f, z] \in \mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$.
2. *For any $z \in \mathbb{F}_q, y \in L_{i+1}$, the function* **Fold** $[f, z]$ *can be evaluated at y by querying f on the roots of the polynomial $q_i(X) - y$.*

Proof. The first item holds by Definition 2.11. For the second item, consider $y \in L_{i+1}$. Denote $S_y \subset L_i$ the set $S_y := q_i^{-1}(\{y\})$. Since $q_i(X) - y$ has two distinct roots, S_y has size 2. Let us consider $P_{f,y} \in \mathbb{F}_q[X]$ the polynomial of degree less than 2 such that, for all $x \in S_y$, $P_{f,y}(x) = f(x)$. We have that the two polynomials $P_{f,y}(X)$ and $\widehat{g_0}(y) + X\widehat{g_0}(y)$ are equal in $\mathbb{F}_q[X]$, since they have both degree less than 2 and agree on two distinct values. Recalling Definition 2.11 we have, for all $z \in \mathbb{F}_q, P_{f,y}(z) =$ **Fold** $[f, z]\,(y)$. In particular, the value **Fold** $[f, z]\,(y)$ can be computed by interpolating the set of points $\{(x, f(x)) \mid x \in S_y\}$ of size two. □

The folding operators defined in Definition 2.11 preserves distance to the code. In order to prove such a property, we state a technical result from [BCI+20]. Proof of Theorem 2.13 uses tools from algebraic geometry and the theory of algebraic function fields. It is proved by running a classical list-decoding algorithm (the Guruswami-Sudan decoder [GS99]) on Reed-Solomon codes whose base field is a rational function field. Note that the following result is specific to Reed-Solomon codes: in contrast to Lemma 1.23, it does not hold for any linear code V.

Theorem 2.13 (Correlated agreement over lines [BCI+20]). *Let $V \coloneqq \mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ be a RS code of rate ρ. Let $u_0, u_1 : L \to \mathbb{F}_q$. Let $\delta, \varepsilon > 0$ satisfy $\varepsilon \leqslant \frac{\sqrt{\rho}}{20}$ and $\delta \leqslant \gamma(\varepsilon, \rho) \coloneqq 1 - \sqrt{\rho} - \varepsilon$, and suppose*

$$\Pr_{z \in \mathbb{F}_q}\left[\Delta\left(u_0 + z u_1, V\right) \leqslant \delta\right] > \frac{k^2}{(2\varepsilon)^7 q}.$$

Then u_0, u_1 are simultaneously δ-close to V, i.e. $\exists v_0, v_1 \in V$ and $T \in L$ such that:

- $|T| \geqslant (1-\delta)\,|L|$,
- $u_{0|T} = v_{0|T}$,
- $u_{1|T} = v_{1|T}$.

From Theorem 2.13, we can prove the following proposition.

Proposition 2.14. *Let ρ be the rate of the code RS code $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < \frac{\sqrt{\rho}}{20}$ and $\delta < 1 - \sqrt{\rho} - \varepsilon$. For any function $f : L_i \to \mathbb{F}_q$ satisfying $\Delta(f, \mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]) > \delta$, we have*

$$\Pr_{z \in \mathbb{F}_q}\left[\Delta(\textbf{Fold}\left[f, z\right], \mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right] \leqslant \delta\right] \leqslant \frac{{k_{i+1}}^2}{(2\varepsilon)^7 q}.$$

Proof. Write $\textbf{Fold}\left[f, z\right] = u_0 + z u_1$ and consider the set

$$A \coloneqq \left\{z \in \mathbb{F}_q \mid \Delta(u_0 + z u_1, \mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]) \leqslant \delta\right\}.$$

We want to show that $|A| \leqslant \frac{{k_{i+1}}^2}{(2\varepsilon)^7}$. By way of contradiction, assume that $|A| > \frac{{k_{i+1}}^2}{(2\varepsilon)^7}$. Thus, by Theorem 2.13, there are $v_0, v_1 \in \mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$ such that the set

$$T \coloneqq \{y \in L_{i+1} \mid u_0(y) = v_0(y) \text{ and } u_1(y) = v_1(y)\}$$

has size $|T| \geqslant (1-\delta)\,|L_{i+1}|$. Let $\widehat{v_0}, \widehat{v_1} \in \mathbb{F}_q[X]_{<k_{i+1}}$ be the polynomials of degree less than k_{i+1} associated to the codewords v_0, v_1, respectively. We have

$$\begin{aligned}\deg\left(\widehat{v_0}(q_i(X)) + X\widehat{v_1}(q_i(X))\right) &\leqslant 2 \cdot (k_{i+1} - 1) + 1 \\ &< k_i.\end{aligned}$$

We deduce that the function $v : L_i \to \mathbb{F}_q$ defined by $v(x) = v_0(q_i(x)) + x v_1(q_i(x))$ for all $x \in L_i$ is a codeword of $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$. For every $y \in T$, consider $x, x' \in L_i$ the two distinct roots of the polynomial $q_i(X) - y$. We have

$$\begin{aligned}v(x) &= v_0(q_i(x)) + x v_1(q_i(x)) \\ &= u_0(q_i(x)) + x u_1(q_i(x)) \\ &= f(x),\end{aligned}$$

and similarly, $v(x') = f(x')$. Therefore

$$\{x \in L_i \mid q_i(x) \in T\} \subseteq \{x \in L_i \mid v(x) = f(x)\}.$$

We deduce that

$$\begin{aligned} |\{x \in L_i \mid v(x) = f(x)\}| &\geqslant |\{x \in L_i \mid q_i(x) \in T\}| \\ &\geqslant 2(1-\delta)\,|L_{i+1}| \\ &\geqslant (1-\delta)\,|L_i|. \end{aligned}$$

Finally, we get that $\Delta(f, v) \leqslant \delta$ and $\Delta(f, \mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]) \leqslant \delta$, which is a contradition. □

Construction 2.15 (FRI protocol [BBHR18a])**.**
Input: *A function $f_0 : L_0 \to \mathbb{F}_q$ (given explicitly to the prover, and as oracle input to the verifier), expected to belong to a RS code* $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]$.

COMMIT ***phase:***

1. *For i from 0 to $\mathsf{r}-2$:*
   (a) *Verifier $\mathscr{V}$ sends an element $z_i \xleftarrow{\$} \mathbb{F}_q$;*
   (b) *Prover $\mathscr{P}$ gives oracle access to $f_{i+1} : L_{i+1} \to \mathbb{F}_q$, supposedly equal to* $\mathbf{Fold}\left[f_i, z_i\right]$.
2. *Prover $\mathscr{P}$ sends a constant $\beta \in \mathbb{F}_q$.*

QUERY ***phase:***

1. *Interpret f_r as the constant function equal to β on L_r.*
2. *Repeat s times:*
   (a) *Sample $y_0 \in L_0$ uniformly at random;*
   (b) *For i from 0 to $\mathsf{r}-1$:*
      i. *Define $y_{i+1} \in L_{i+1}$ as $y_{i+1} = q_i(y_i)$;*
      ii. *Query f_i on the roots of the polynomial $q_i(X) - y_{i+1}$ to compute* $\mathbf{Fold}\left[f_i, z_i\right](y_{i+1})$;
      iii. *Query $f_{i+1}(y_{i+1})$;*
      iv. *If $f_{i+1}(y_{i+1}) \neq \mathbf{Fold}\left[f_i, z_i\right](y_{i+1})$, outputs **reject*** (Round consistency check)*;*
3. *Outputs **acccept***

Soundnesss analysis in outline. Let us give some intuition about the soundness of the FRI protocol. Assume that f_0 is δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]$. If a prover $\widetilde{\mathscr{P}}$ compute its oracle message as prescribed by the protocol, namely by sending

$$f_{i+1} = \mathbf{Fold}\left[f_i, z_i\right]$$

at each round $i \in [\![0, r-1]\!]$, then each oracle function f_{i+1} is also δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$ (with high probability). In particular, assuming that the folding operation has preserved the distance to the code at each round, the last function

$$f_r = \mathbf{Fold}\left[f_{r-1}, z_{r-1}\right]$$

is at distance at least δ from the last code $\mathsf{RS}\left[\mathbb{F}_q, L_r, k_r\right]$. In this case, one can prove that the verifier $\mathscr{V}$ accepts with probability at most $1-\delta$.

As a result, a malicious prover $\widetilde{\mathscr{P}}$ can be tempted to cheat at some round before the last one, by sending a function

$$f_{i+1} \neq \textbf{Fold}\left[f_i, z_i\right].$$

Essentially, a verifier can be deceived in two ways:

- the verifier $\mathscr{V}$ may randomly sample a "bad" challenge, namely for some round $i \in [\![0, r-1]\!]$, $\mathscr{V}$ picks an element $z_i \in \mathbb{F}_q$ such that the distance of the function $\textbf{Fold}\left[f_i, z_i\right]$ to $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$ is significantly smaller than the distance of f_i to $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$;
- or the malicious prover $\widetilde{\mathscr{P}}$ sends an oracle function $f_{i+1} \neq \textbf{Fold}\left[f_i, z_i\right]$ whose distance to $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$ is significantly smaller than the distance of f_i to $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$.

By Proposition 2.14 and a union bound, the probability that the event described in the first item happens is bounded from above by

$$\mathsf{err}_{\mathsf{commit}} = r \cdot \frac{k^2}{(2\varepsilon)^7 q}.$$

Regarding the second item, the basic idea is that the more the function f_{i+1} sent by $\widetilde{\mathscr{P}}$ differs from $\textbf{Fold}\left[f_i, z_i\right]$, the higher the probability that the round consistency check fails. Soundness analysis consists in proving that whenever a prover tries to "correct errors" and sends an oracle function f_{i+1} which is closer to the code $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$ (compared to the distance of $\textbf{Fold}\left[f_i, z_i\right]$ to $\mathsf{RS}\left[\mathbb{F}_q, L_{i+1}, k_{i+1}\right]$), this is foiled by an increased probability of failing the round consistency test. Recall that the verifier rejects if, for some $i \in [\![0, r-1]\!]$, the verifier inspects a location $y \in L_{i+1}$ such that

$$f_{i+1}(y) \neq \textbf{Fold}\left[f_i, z_i\right](y).$$

The main and most intricate part of the soundness analysis consists in estimating the probability that at least one round consistency test fails (assuming that the event of the first item has not occurred during the COMMIT phase). It turns out that, assuming that no bad challenge has been sent (which happens with probability at least $1 - \mathsf{err}_{\mathsf{commit}}$), the probability that the verifier accepts during a single repetition of the QUERY phase is at most $\mathsf{err}_{\mathsf{query}} = 1 - \delta$.

The first analysis was given by [BBHR18a] and relied on [PS94, BS08]. Soundness of the FRI protocol was further improved over several subsequent works [BKS18, BGKS20, BCI+20]. The complete and formal soundness analysis of the FRI protocol is indeed technically involved. Since we will provide a soundness analysis for a more generic setting in Chapter 3, we only state below the result from [BCI+20, Section 8], which is a consequence of Theorem 2.13 (which is also a result from [BCI+20]). Alternatively, the following theorem can be proved by replacing [BKS18, Corollary 7.3] with Proposition 2.14 in the proof of [BKS18, Theorem 7.2].

Theorem 2.16 ([BCI+20]). *Let ρ denote the rate of $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$. Let $\delta, \varepsilon > 0$ satisfy $\varepsilon \leqslant \frac{\sqrt{\rho}}{20}$ and $\delta \leqslant \gamma(\varepsilon, \rho) \coloneqq 1 - \sqrt{\rho} - \varepsilon$. For any function f which is δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ and unbounded prover $\widetilde{\mathscr{P}}$, the verifier $\mathscr{V}_{\mathsf{RS}}$ accepts after α repetitions of the QUERY phase with probability at most*

$$\mathsf{err}_{commit} + (\mathsf{err}_{query})^{\alpha},$$

where

$$\mathsf{err}_{commit} = \frac{k^2}{(2\varepsilon)^7 q} \log k \qquad \text{and} \qquad \mathsf{err}_{query} = (1 - \delta).$$

The main properties of the FRI protocol are stated below. Constants appearing in Theorem 2.17 are different than [BBHR18a] for the reasons mentioned in Remark 2.10.

Theorem 2.17 (Properties of FRI). *Assume* $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$ *is a RS code where* k *is a power of 2 and* L *is a coset of either a multiplicative or an additive subgroup defined as per Section 2.1.4. Construction 2.15 is a public-coin IOPP* $(\mathscr{P}_{\mathsf{RS}}, \mathscr{V}_{\mathsf{RS}})$ *satisfying perfect completeness and soundness as stated in Theorem 2.16. Moreover, the IOPP system* $(\mathscr{P}_{\mathsf{RS}}, \mathscr{V}_{\mathsf{RS}})$ *has the following properties:*

- *rounds complexity* $\mathsf{r} = \log k$,
- *proof length* $\mathsf{l} < n$,
- *query complexity* $\mathsf{q} = 2\alpha \log k + 1$,
- *prover complexity* $\mathsf{tp} < 8n$,
- *verifier complexity* $\mathsf{tv} < 8\alpha \log k$,

Proof of Theorem 2.17. The completeness follows from the fact that, if f_0 is a codeword, then Definition 2.11 directly implies that every honestly computed function f_i is a codeword of $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$. Since $k_{i+1} = k_i/2$, the dimension k_r of the last RS code $\mathsf{RS}\left[\mathbb{F}_q, L_r, k_r\right]$ is 1. Every round consistency test passes and f_r is a constant function equal to β. Thus, the verifier always accepts.

For the rest of the proof, we adapt the one of [BBHR18a] to the presented setting. By construction, the IOPP has $\mathsf{r} = \log k$ rounds. Let us denote by n_i the length of the code $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]$ for $i \in [\![0, r]\!]$. The proof length is sum of the n_i's for $i \in [\![1, r]\!]$. Since $n_{i+1} = n_i/2$, the sum of the first terms of a geometric sequence gives the claimed proof length.

During the QUERY phase, the verifier makes 2 queries to each function $f_0, \ldots, f_{r-1}$. The verifier also queries the element β sent during the last round of the COMMIT phase. This gives the claimed query complexity.

Given $y \in q(L)$ and $S_y \coloneqq q^{-1}(\{y\})$, we compute the cost c of evaluating $\mathbf{Fold}\left[f, z\right]$ at y. Recall that $\mathbf{Fold}\left[f, z\right](y)$ can be computed by interpolating the set of points $\{(x, f(x)) \mid q_i(x) = y\}$ and evaluating the obtained polynomial at z (see proof of Lemma 2.12).

Let us consider x, x' the roots of the polynomial $q_i(X) - y$. We have

$$\mathbf{Fold}\left[f, z\right](y) = P_{f,y}(z) = \frac{1}{x - x'}\left(f(x)(z - x) - f(x')(z - x')\right).$$

Therefore, computing $\mathbf{Fold}\left[f, z\right](y)$ takes at most 8 field operations.

At each round, the prover performs $8n_i$ computations. Summing over r rounds, we get the claimed prover complexity. During one round of the QUERY phase, the verifier evaluates $\mathbf{Fold}\left[f, z\right]$ at a single point, therefore the verifier complexity for a repetition parameter α is $8\alpha r \leqslant 8\alpha m \log k$. □

How to choose the repetition parameter. Assuming that $\mathsf{err}_{\mathsf{commit}} < 1$ in Theorem 2.17, repeating the COMMIT phase enough times allows to get $\mathsf{err}_{\mathsf{commit}} < 2^{-\kappa}$ for any security parameter κ. By letting $\delta \coloneqq 1 - \sqrt{\rho} - \varepsilon$ in Theorem 2.17, a repetition parameter α in the QUERY phase such that

$$\alpha > -\frac{\kappa}{\log\left(\sqrt{\rho} + \varepsilon\right)}$$

yields

$$\mathsf{err}_{\mathsf{query}}{}^{\alpha} = (1 - \delta)^{\alpha} < 2^{-\kappa},$$

Note that the number of repetitions of the QUERY phase decreases with the code rate.

Additional considerations on parameters. In practice, the size of the finite field can be chosen as a function of a security parameter κ, typically $|\mathbb{F}_q| > 2^\kappa$. In that case, the domain size $|L|$ is generally several orders of magnitude smaller than $|\mathbb{F}_q|$. However, it is also possible to reach soundness error less than $2^{-\kappa}$ even for fields of size much smaller than 2^κ, by repeating the interactive phase (the so-called COMMIT phase) of the protocol.

The domain size $|L|$ is a lower bound on the prover running time of the FRI protocol. Thus, the prover runtime benefits from a larger code rate ρ. On the other hand, in order to reduce the number of repetitions of the QUERY phase and thus reduce the total query complexity, it is desirable to lower the code rate ρ (or, equivalently, consider Reed-Solomon codes with a higher minimum distance). As mentioned before, reducing query complexity also reduces the size of succinct non-interactive arguments obtained from interactive oracle proofs.

In summary, the choice of the code rate ρ gives a trade-off between prover time and size of non-interactive arguments based on the FRI protocol.

On improving the soundness of the FRI protocol. Observe that, for the soundness error to be non trivial in Theorem 2.16, the size of the field must be quadratic in the blocklength of the code. Soundness analyses for field of linear size appeared in [BBHR18a, BKS18, BGKS20], but the bound $\gamma(\varepsilon,\rho)$ on the proximity parameter δ in soundness of Theorem 2.17 was strictly smaller, for any rate. In practice, the larger is the bound on δ, the smaller can be set the repetition parameter α to reach soundness error less than $2^{-\kappa}$, for some security parameter κ. Since the total query complexity is the dominant factor in the size of non-interactive argument based on IOPs (see Section 1.2.6), lowering the repetition parameter α is of significant importance for practical implementations.

The authors of [BBHR18a] proposed a conjecture stating that the soundness error given in Theorem 2.17 holds for δ as large as $\approx 1-\rho$, rather than $\approx 1-\sqrt{\rho}$ (see [BBHR18a, BCI+20]). Assuming $\gamma(\varepsilon,\rho) \approx 1-\rho$ instead of $\gamma(\varepsilon,\rho) \approx 1-\sqrt{\rho}$ in Theorem 2.17 enables to roughly halve the number of repetitions α of the QUERY phase when targeting soundness error less than $2^{-\kappa}$. Accordingly, query complexity and verifier complexity are also divided by approximately 2.

Determining the optimal value of $\gamma(\varepsilon,\rho)$ in Theorem 2.13 and the minimum number of elements $z \in \mathbb{F}_q$ such that such a statement holds remains an open problem with intriguing relation to coding theory. Besides, considering its relation with the soundness error of the FRI protocol, it has also significant consequences in the concrete efficiency of the FRI protocol in practice. We finally note that practical implementations of the FRI protocol rely on this conjecture [BBHR19, BCR+19].

Remark 2.18. *There are IOPs of Proximity with constant-query complexity (instead of logarithmic query complexity) that also solve the Reed-Solomon proximity problem [BCG+17, RR20], but at the cost of inefficient prover and verifier algorithms. In constrast, the prover and the verifier of the FRI protocol are highly efficient (see Theorem 2.17), which is crucial for real-world applications and explains why the FRI protocol is the solution that is implemented and used in practice [BBHR19, Sta].*

2.2 An IOP-based SNARG using Reed-Solomon proximity testing

The goal of this section is to illustrate how computational integrity statements can be reduced to RS proximity testing.

Recall that $\mathbf{NTIME}(T(n))$ is the set of languages that have membership proofs (i.e. nondeterministic witness) verifiable in time $O(T(n))$ by a deterministic machine M (Definition 1.2). Given a purported instance x of a language $\mathcal{L} \in \mathbf{NTIME}(T(n))$, a SNARG is a $o(T(n))$-size proof showing that there exists w such that $M(\mathsf{x},\mathsf{w}) = 1$. Ideally, the size and verification time of a SNARG are much smaller than the naive solution where the prover sends w and the verifier runs $M(\mathsf{x},\mathsf{w})$. However, note that sublinear verification (without preprocessing of the instance) is possible only if the instance admits a succinct representation, namely much smaller than $T(n)$.

The "Stark" construction is a SNARG introduced in [BBHR18b] which relies on Reed-Solomon proximity testing. For a language $\mathcal{L} \in \mathbf{NTIME}(T(n))$, a Stark proof with soundness error $2^{-\kappa}$ has size $O(\kappa^2 \log^2 T(n))$, can be generated in time $O(\kappa^2 T(n) \log^2 T(n))$ and verified in time $\widetilde{O}(n) + O(\kappa^2 \log^2 T(n))$. The name of the construction stands for *Scalable Transparent ARgument of Knowledge*, emphasizing that a Stark proof is a succinct non-interactive argument of knowledge, with transparent setup, fast proving time and fast verification (hence "scalable").

While the construction of [BBHR19] is valid for any languages in **NP** and **NEXP**, its interest is best understood when having in mind the computational integrity language $\mathcal{L}_{\mathsf{CI}}$ (see e.g. [BBHR18b, Definition 3.1]), which is **NEXP**-complete and consists of tuples (P, x, y, T) such that the program P on input x reaches result y in at most T computational steps. Indeed, the Stark proof system achieve polylogarithmic verification for uniform computations represented by a sequential program P and a time bound T, where T is typically much greater than the description of P. Note that, for instance, if the program P represents the execution of a generic n-gate arithmetic circuit, then $|P|, T = \Omega(n)$. Thus in that case, checking a Stark proof would be no faster than naive verification.

The Stark construction is designed for a specific **NEXP**-complete language which facilitates the construction of the proof system, called *Algebraic Intermediate Representation* (AIR). Informally, instances of the AIR language consist in a set of polynomial constraints specifying the transition function of a RAM program. An AIR instance belong to the AIR language if there exists an "execution trace of the program" (the nondeterministic witness) that satisfies the prescribed constraints. Therefore, a Stark prover shows that *there exists* a satisfying witness for a given AIR instance (a prover actually shows that *he knows* such a witness, but we will not address this property in our discussion). As for any proof system, it is assumed that the conversion from a computational integrity statement of the form "a given program P outputs y on input x within T steps" to an equivalent instance of the AIR language has been done beforehand.

Arithmetization reduces instances of the AIR language to instances of the Reed-Solomon proximity testing problem. The FRI protocol discussed in Section 2.1.5 is used as a solution of the latter problem, and the efficiency parameters of the Stark construction crucially rely on it.

The running time of the prover is dominated by the time of encoding Reed-Solomon codewords over evaluation domains of size $\Theta(T)$. Therefore, the field is chosen such that RS encodings can be performed with $O(T \log T)$ field operations, using FFT or additive FFT [LANHC16]. It is also the setting required for the FRI protocol to work (see Section 2.1.4).

Once an IOP for the AIR language is constructed, it is compiled into a SNARG with unconditional security in the quantum random oracle model [CMS19] using the BCS transformation [BCS16].

Since the construction of SNARGs is not the main topic of this thesis, we will only present a simplified version of the IOP protocol of [BBHR19]. A complete description of the protocol can be found in [BBHR18b], and a variant is presented [Sta21a, Section 5.3]. Our presentation is adapted from those two references. We start by defining the AIR language for space-bounded computations, then we construct an IOP protocol for it.

2.2.1 Algebraic Intermediate Representation

Following terminology introduced in [BBHR18b], we are interested in computations presented in a form called *algebraic intermediate representation* (AIR). We focus on computations that are abstractly executed over a machine whose full state can be captured by a fixed number w of field elements (a treatment of general sequential computations can be found in [BBHR18b]). The program for which we want to verify the correct execution is assumed to be executed over an algebraic machine, meaning a machine which naturally operates over finite field elements.

Informally, an AIR instance specifies a computation by defining a finite set $\mathcal{C}$ of "constraint polynomials" in $\mathbb{F}_q[Y_1, \ldots, Y_w, Y'_1, \ldots, Y'_w]$, as well as a set of boundary constraints $\mathcal{B} \subset [\![1, w]\!] \times [\![0, T]\!] \times \mathbb{F}_q$. An execution trace is a $(T+1) \times w$ matrix of field elements. One can think of the w columns of this matrix as the contents of w registers $p_1, \ldots, p_w$ over time, while rows correspond to the successive machine states. Roughly speaking, an AIR instance will be satisfied by $p_1, \ldots, p_w \in \mathbb{F}^{T+1}$ if the following conditions are satisfied:

i. each pair of consecutive states $(\boldsymbol{y}, \boldsymbol{y}') \in \mathbb{F}_q^w \times \mathbb{F}_q^w$ is a valid transition, i.e. $(\boldsymbol{y}, \boldsymbol{y}')$ is a common zero of the polynomials in $\mathcal{C}$;
ii. for each boundary constraints $(i, t, \alpha) \in \mathcal{B}$, where $i \in [\![1, w]\!]$ is a register index, $t \in [\![0, T]\!]$ a timestamp and $\alpha \in \mathbb{F}_q$, $p_i(t) = \alpha$.

For the sake of this presentation, we focus on the case where $\mathbb{F}_q$ is a prime field of size $|\mathbb{F}_q| = \Omega(T)$. In this setting, the range $[\![0, T]\!]$ is identified with a multiplicative subgroup $H = \langle g \rangle \subseteq \mathbb{F}_q$ such that $|H| = T + 1$. Accordingly, an AIR assignment will be composed of w functions in $\mathbb{F}_q^H$, instead of w vectors in $\mathbb{F}^{T+1}$. Note that H has a succinct representation, due to its algebraic structure: $O(\log |\mathbb{F}_q|)$ bits are enough, while a generic set would be represented with $O(T \log |\mathbb{F}_q|)$ bits. Another crucial fact is that the vanishing polynomial of H is $Z_H(X) = X^{T+1} - 1$. Consequently, it can be evaluated by the verifier at any single point in $O(\log T)$ field operations using exponentiation by squaring.

Definition 2.19 (AIR instance)**.** *An algebraic intermediate representation (AIR) instance is a tuple* $\mathsf{x}_{\mathsf{AIR}} = (\mathbb{F}_q, g, w, T, d_{\mathcal{C}}, s, \mathcal{C}, \mathcal{B})$ *where:*

- $\mathbb{F}_q$ *is a finite field;*
- w, h, d, s *are integers indicating the following sizes:*
  - w *is the number of columns in the trace,*
  - T *is a time bound,*
  - d *is the maximal degree of a constraint,*
  - s *is the size of the set of constraints;*
- g *is a generator of multiplicative subgroup* $H \subset \mathbb{F}_q^\times$ *of order* $T+1$ *(*H *is called the* trace domain*);*
- $\mathcal{C} = \{Q_1, \ldots, Q_s\}$ *is a finite set of constraint polynomials, where* $Q_i \in \mathbb{F}_q[Y_1, \ldots, Y_w, Y'_1, \ldots, Y'_w]$ *is a polynomial of total degree at most* $d_{\mathcal{C}}$*,*

– $\mathcal{B} \subset [\![1, w]\!] \times [\![0, T]\!] \times \mathbb{F}_q$ *is a finite set of boundary constraints.*

Definition 2.20 (AIR assignment)**.** *An AIR assignment is a tuple* $\boldsymbol{p} = (p_1, \ldots, p_w)$ *where* $p_i : H \to \mathbb{F}_q$.

Definition 2.21 (Composition polynomial)**.** *Given an AIR constraint polynomial* $Q \in \mathcal{C}$ *of an AIR instance* $\mathsf{x}_{\mathsf{AIR}} = (\mathbb{F}_q, g, w, T, d, s, \mathcal{C}, \mathcal{B})$ *and* $\boldsymbol{P} = (P_1, \ldots, P_w)$ *a tuple of polynomials in* $\mathbb{F}_q[X]$*, the* composition polynomial *of* Q *and* $\boldsymbol{P}$ *is the univariate polynomial denoted* $Q \circ \boldsymbol{P}$ *and defined as*

$$(Q \circ \boldsymbol{P})(X) \coloneqq Q(P_1(X), \ldots, P_w(X), P_1(gX), \ldots, P_w(gX)) \in \mathbb{F}_q[X].$$

Definition 2.22 (AIR satisfiability)**.** *Given an AIR assignment* $\boldsymbol{p} = (p_1, \ldots, p_w)$*, we associate a tuple* $\boldsymbol{P} = (P_1, \ldots, P_w)$ *of polynomials in* $\mathbb{F}_q[X]$ *where* P_i *is the polynomial of degree less than* $|H|$ *such that* $P_i(g^t) = p_i(g^t)$ *for all* $t \in [\![0, |H| - 1]\!]$.

An AIR assignment $\boldsymbol{p}$ *is said to* satisfy *an AIR instance* $\mathsf{x}_{\mathsf{AIR}}$ *if and only if the two following conditions hold:*

1. *for every* $(i, t, \alpha) \in \mathcal{B}$*,* $P_i(g^t) = \alpha$.
2. *for every* $Q_i \in \mathcal{C}$*, the composition polynomial* $Q_i \circ \boldsymbol{P}$ *vanishes on* $H \setminus \{g^T\}$.

We say that $\mathsf{x}_{\mathsf{AIR}}$ *is* satisfiable *if there exists an AIR assigment* $\boldsymbol{p}$ *that satisfies it.*

Definition 2.23. *The AIR language* $\mathcal{L}_{\mathsf{AIR}}$ *is the set of satisfiable AIR instances.*

2.2.2 A simple IOP for the AIR language

We now present an IOP protocol enabling a prover $\mathscr{P}$ to prove that there exists an AIR assignment $\boldsymbol{p}$ satisfying a given instance $\mathsf{x}_{\mathsf{AIR}}$.

The main idea is to reduce the problem of testing whether a univariate polynomial vanishes on a given subset to a univariate low-degree test. Observe that for an AIR assignement which satisfies Item 2 of Definition 2.22, the polynomial $\prod_{t=0}^{T-1}(X - g^t) = \frac{Z_H(X)}{X - g^T}$ divides $(Q_i \circ \boldsymbol{P})(X)$ for every constraint polynomial in $\mathcal{C}$. This means that the rational function

$$\frac{(Q_i \circ \boldsymbol{P})(X)}{\prod_{t=0}^{T-1}(X - g^t)}$$

is in fact a univariate polynomial of degree less than $d(|H| - 1) - |H| + 2$. The satisfaction of a boundary constraints will also be translated in term of divisibility of univariate polynomials.

Construction 2.24 (IOP for $\mathcal{L}_{\mathsf{AIR}}$)**.**
Let $(\mathscr{P}_{\mathsf{RS}}, \mathscr{V}_{\mathsf{RS}})$ *be an IOPP system for the relation* $\mathcal{R}_{RS}$ *with soundness error* $\mathsf{err}_{\mathsf{LDT}} : [0, 1] \to [0, 1]$.
Inputs: *Prover* $\mathscr{P}$ *and Verifier* $\mathscr{V}$ *receive as inputs an AIR instance* $\mathsf{x}_{\mathsf{AIR}} = (\mathbb{F}_q, g, w, T, d, s, \mathcal{C}, \mathcal{B})$. *Prover* $\mathscr{P}$ *is given an AIR assigment* $\boldsymbol{p} \in \left(\mathbb{F}_q^H\right)^w$.

Additionally, they are both given an RS evaluation domain $L \subset \mathbb{F}_q$ *such that* $L \cap H = \varnothing$ *and* $|L| > \max\{|H|, d(|H| - 1) - |H| + 2\}$ *and a parameter* $\delta \in (0, 1)$ *such that*

$$\delta < \min\left\{\frac{1}{2}\left(1 - \frac{|H|}{|L|}\right), \frac{1}{2w + 1}\left(1 - \frac{d(|H| - 1) + 2}{|L|}\right)\right\}.$$

Common definitions: *For every* $i \in [\![1, w]\!]$ *such that there exists a tuple* $(i, t, \alpha) \in \mathcal{B}$*, define:*

- $B_i \in \mathbb{F}_q[X]$ *the polynomial of minimal degree such that* $B_i(g^t) = \alpha$ *for all* $(i,t,\alpha) \in \mathcal{B}$*;*
- $Z_i \in \mathbb{F}_q[X]$ *the polynomial vanishing on the locations where the boundary constraints must be satisfied, namely*

$$Z_i(X) \coloneqq \prod_{t\colon (i,t,\alpha)\in\mathcal{B}} (X - g^t)$$

of degree $\deg Z_i = |\{t\colon (i,t,\alpha) \in \mathcal{B}\}|$.

Moreover, denote $Z_H \in \mathbb{F}_q[X]$ *the vanishing polynomial of* H.
Protocol:

1. Execution trace oracles: *For each* $i \in [\![1,w]\!]$,
   (a) *Prover* $\mathcal{P}$ *interpolates the polynomial* $P_i \in \mathbb{F}_q[X]_{<|H|}$ *such that* $P_i(g^t) = p_i(g^t)$ *for all* $t \in [\![0,|H|-1]\!]$.
   (b) *Prover* $\mathcal{P}$ *gives oracle access to* $f_i : L \to \mathbb{F}_q$*, where* f_i *is the evaluation of* P_i *on* L.
2. Boundary oracles: *Prover* $\mathcal{P}$ *gives oracle access to* w *functions* $h_1,\dots,h_w : L \to \mathbb{F}_q$*, where* h_i *is the evaluation on* L *of the rational function*

$$\frac{P_i(X) - B_i(X)}{Z_i(X)}. \tag{2.2}$$

3. Constraint oracles: *Prover* $\mathcal{P}$ *gives oracle access to* s *functions* $g_1,\dots,g_s : L \to \mathbb{F}_q$*, where* g_i *is the evaluation on* L *of the rational function*

$$\frac{X - g^T}{Z_H(X)}(Q_i \circ \boldsymbol{P})(X). \tag{2.3}$$

4. Consistency checks: *Verifier* $\mathcal{V}$ *samples* $\alpha \in L$ *and check the following equations by querying* $f_1,\dots,f_w,h_1,\dots,h_w,g_1,\dots,g_s$ *for the needed values:*
   (a) *for all* $i \in [\![1,w]\!]$, $Z_i(\alpha)h_i(\alpha) \stackrel{?}{=} f_i(\alpha) - B_i(\alpha)$*;*
   (b) *for all* $j \in [\![1,s]\!]$, $Z_H(\alpha)g_j(\alpha) \stackrel{?}{=} (\alpha - g^T)Q_j(f_1(\alpha),\dots,f_w(\alpha),f_1(g\alpha),\dots,f_w(g\alpha))$.
5. RS proximity tests: *Run* $(\mathcal{P}_{\mathsf{RS}},\mathcal{V}_{\mathsf{RS}})$ *for the following proximity tests:*
   (a) *For each* $i \in [\![1,w]\!]$*, check* δ*-proximity of* f_i *to* $\mathsf{RS}\left[\mathbb{F}_q, L, |H|\right]$*;*
   (b) *For each* $i \in [\![1,w]\!]$*, check* δ*-proximity of* h_i *to* $\mathsf{RS}\left[\mathbb{F}_q, L, |H| - \deg Z_i\right]$*;*
   (c) *For each* $j \in [\![1,s]\!]$*, check* δ*-proximity of* g_j *to* $\mathsf{RS}\left[\mathbb{F}_q, L, d(|H|-1) - |H| + 2\right]$.
6. Decision: *Verifier* $\mathcal{V}$ *accepts if and only if (i) all consistency checks pass at Step 4 and (ii) for every proximity test performed at Step 5,* $\mathcal{V}_{\mathsf{RS}}$ *accepts.*

Proposition 2.25. *Construction 2.24 is an IOP for* $\mathcal{L}_{\mathsf{AIR}}$ *which satisfies the following properties.*

Completeness: *If* $\mathsf{x}_{\mathsf{AIR}}$ *is satisfied by* $\boldsymbol{p}$*, then the verifier* $\mathcal{V}$ *always accepts after interacting with the prover* $\mathcal{P}$.

Soundness: *If* $\mathsf{x}_{\mathsf{AIR}} \notin \mathcal{L}_{\mathsf{AIR}}$*, then for any prover strategy* $\widetilde{\mathcal{P}}$*, the verifier* $\mathcal{V}$ *accepts with probability at most*

$$\max\left\{\mathsf{err}_{\mathsf{LDT}}(\delta), \frac{|H|}{|L|} + 2\delta, \frac{d(|H|-1)+2}{|L|} + (2w+1)\delta\right\},$$

where δ *is part of the inputs of Construction 2.24.*

Observe that δ is set in Construction 2.24 so that $\frac{|H|}{|L|} + 2\delta$ and $\frac{d(|H|-1)+2}{|L|} + (2w+1)\delta$ are smaller than 1 .

Proof. Suppose that $\boldsymbol{p}$ is a satisfying AIR assignment for $\mathsf{x}_{\mathsf{AIR}}$. Then $P_i(g^t) = B_i(g^t)$ for all $(i,t,\alpha) \in \mathcal{B}$, and thus the rational function (2.2) is in fact a polynomial of degree less than $|H| - \deg Z_i$. Moreover, the expression (2.3) is also a polynomial, with degree less than $d(|H|-1) - |H| + 2$. Therefore, the prover can make all the consistency checks and proximity tests pass, and the verifier $\mathscr{V}$ accepts by completeness of $(\mathscr{P}_{\mathsf{RS}}, \mathscr{V}_{\mathsf{RS}})$. This proves completeness.

Regarding soundness, we proceed by contraposition. Let us assume that the verifier accepts with probability greater than

$$\max\left\{\mathsf{err}_{\mathsf{LDT}}(\delta), \frac{|H|}{|L|} + 2\delta, \frac{d(|H|-1)+2}{|L|} + (2w+1)\delta\right\}.$$

We will show that $\mathsf{x}_{\mathsf{AIR}} \in \mathcal{L}_{\mathsf{AIR}}$ by constructing a satisfying assignment for $\mathsf{x}_{\mathsf{AIR}}$.

First, since the verifier accepts with probability greater than $\mathsf{err}_{\mathsf{LDT}}(\delta)$, we have:

i. for all $i \in [\![1,w]\!]$, $\Delta\left(f_i, \mathsf{RS}\left[\mathbb{F}_q, L, |H|\right]\right), \Delta\left(h_i, \mathsf{RS}\left[\mathbb{F}_q, L, |H| - \deg Z_i\right]\right) \leqslant \delta$ and,
ii. for all $j \in [\![1,s]\!]$, $\Delta\left(g_j, \mathsf{RS}\left[\mathbb{F}_q, L, d(|H|-1) - |H| + 2\right]\right) \leqslant \delta$.

Therefore, we can consider $c_{f,i} \in \mathsf{RS}\left[\mathbb{F}_q, L, |H|\right]$ (resp. $c_{h,i} \in \mathsf{RS}\left[\mathbb{F}_q, L, |H| - \deg Z_i\right]$) such that $\Delta(c_{f,i}, f_i) \leqslant \delta$ (resp. $\Delta(c_{h,i}, h_i) \leqslant \delta$) for each $i \in [\![1,w]\!]$. Similarly, for each $j \in [\![1,s]\!]$, let

$$c_{g,j} \in \mathsf{RS}\left[\mathbb{F}_q, L, d(|H|-1) - |H| + 2\right]$$

be such that $\Delta(c_{g,j}, g_j) \leqslant \delta$.

Observe that, for the verifier to accept, all consistency tests at Step 4 must succeed. Therefore, for every $i \in [\![1,w]\!]$, we have

$$\Pr_{\alpha \in L}\left[Z_i(\alpha)h_i(\alpha) = f_i(\alpha) - B_i(\alpha)\right] > \frac{|H|}{|L|} + 2\delta.$$

Let us fix $i \in [\![1,w]\!]$. We also have

$$\begin{aligned}\Pr_{\alpha \in L}[Z_i(\alpha)h_i(\alpha) = f_i(\alpha) - B_i(\alpha)] \leqslant \Pr_{\alpha \in L}[Z_i(\alpha)h_i(\alpha) = f_i(\alpha) - B_i(\alpha) \mid f_i(\alpha) = c_{f,i}(\alpha) \wedge h_i(\alpha) = c_{h,i}(\alpha)] \\ + \Pr_{\alpha \in L}[f_i(\alpha) \neq c_{f,i}(\alpha)] + \Pr_{\alpha \in L}[h_i(\alpha) \neq c_{h,i}(\alpha)].\end{aligned}$$

Combining the two inequalities above and recalling the definitions of $c_{f,i}$ and $c_{h,i}$, we get

$$\Pr_{\alpha \in L}[Z_i(\alpha)h_i(\alpha) = f_i(\alpha) - B_i(\alpha) \mid f_i(\alpha) = c_{f,i}(\alpha) \wedge h_i(\alpha) = c_{h,i}(\alpha)] > \frac{|H|}{|L|}. \tag{2.4}$$

Let us consider the polynomials $\widehat{c_{f,i}} \in \mathbb{F}[X]_{<|H|}$ and $\widehat{c_{h,i}} \in \mathbb{F}[X]_{<|H|-\deg Z_i}$ whose evaluations on L agree with the codewords $c_{f,i}$ and $c_{h,i}$, respectively. From (2.4), we get

$$\Pr_{\alpha \in L}\left[Z_i(\alpha)\widehat{c_{h,i}}(\alpha) - \widehat{c_{f,i}}(\alpha) + B_i(\alpha) = 0\right] > \frac{|H|}{|L|}.$$

Since $\deg\left(Z_i(X)\widehat{c_{h,i}}(X) - \widehat{c_{f,i}}(X) + B_i(X)\right) < |H|$, we deduce from Corollary 1.20 that

$$Z_i(X)\widehat{c_{h,i}}(X) = \widehat{c_{f,i}}(X) - B_i(X),$$

and thus $Z_i(X)$ divides $\widehat{c_{f,i}}(X) - B_i(X)$. Therefore, for all $(i,t,\alpha) \in \mathcal{B}$, $\widehat{c_{f,i}}(g^t) = B_i(g^t) = \alpha$.

Let us denote by $p'_1, \ldots, p'_w$ the evaluations on H of the polynomials $\widehat{c_{f,1}}, \ldots, \widehat{c_{f,w}}$, respectively. Then $\boldsymbol{p}' = (p'_1, \ldots, p'_w)$ satisfies Item 1 of Definition 2.22. To show that $\boldsymbol{p}'$ is a satisfying assigment for $\mathsf{x}_{\mathsf{AIR}}$, it remains to prove that $\boldsymbol{p}'$ also satisfies Item 2 of Definition 2.22. We proceed similarly.

For $j \in [\![1,s]\!]$, let us denote $E_j(\alpha)$ the event

$$E_j(\alpha) : \text{“}Z_H(\alpha)g_j(\alpha) = (\alpha - g^T)Q_j(f_1(\alpha), \ldots, f_w(\alpha), f_1(g\alpha), \ldots, f_w(g\alpha))\text{”}.$$

For all $j \in [\![1,s]\!]$, we have both

$$\Pr_{\alpha \in L}\left[E_j(\alpha)\right] > \frac{d(|H|-1)+2}{|L|} + (2w+1)\delta$$

and

$$\begin{aligned}\Pr_{\alpha\in L}[E_j(\alpha)] \leqslant \Pr_{\alpha\in L}[E_j(\alpha) \mid g_j(\alpha) = c_{g,j}(\alpha) \wedge \left(\forall i \in [\![1,w]\!], f_i(\alpha) = c_{f,i}(\alpha) \wedge f_i(g\alpha) = c_{f,i}(g\alpha)\right)] \\ + \Pr_{\alpha\in L}[g_j(\alpha) \neq c_{g,j}(\alpha)] + \sum_{i=1}^{w}\Pr_{\alpha\in L}[f_i(\alpha) \neq c_{f,i}(\alpha)] + \sum_{i=1}^{w}\Pr_{\alpha\in L}[f_i(g\alpha) \neq c_{f,i}(g\alpha)].\end{aligned}$$

For $j \in [\![1,s]\!]$, consider $\widehat{c_{g,j}} \in \mathbb{F}[X]$ the polynomial of degree less than $d(|H|-1) - |H| + 2$ which agrees with $c_{g,j}$ on L. We deduce that

$$\Pr_{\alpha\in L}\left[Z_H(\alpha)\widehat{c_{g,j}}(\alpha) - (\alpha - g^T)Q_j(\widehat{c_{f,1}}(\alpha), \ldots, \widehat{c_{f,w}}(\alpha), \widehat{c_{f,1}}(g\alpha), \ldots, \widehat{c_{f,w}}(g\alpha)) = 0\right] > \frac{d(|H|-1)+2}{|L|}.$$

Since the polynomial

$$Z_H(X)\widehat{c_{g,j}}(X) - (X - g^T)Q_j(\widehat{c_{f,1}}(X), \ldots, \widehat{c_{f,w}}(X), \widehat{c_{f,1}}(gX), \ldots, \widehat{c_{f,w}}(gX))$$

has degree less than $d(|H|-1)+2$, it implies that $\frac{Z_H(X)}{X-g^T}$ divides the polynomial

$$Q_j(\widehat{c_{f,1}}(X), \ldots, \widehat{c_{f,w}}(X), \widehat{c_{f,1}}(gX), \ldots, \widehat{c_{f,w}}(gX)).$$

Therefore, $Q_j(\widehat{c_{f,1}}(X), \ldots, \widehat{c_{f,w}}(X), \widehat{c_{f,1}}(gX), \ldots, \widehat{c_{f,w}}(gX))$ vanishes on $H \setminus \{g^T\}$.

We conclude that $\boldsymbol{p}'$ satisfies Item 2 of Definition 2.22, and thus $\boldsymbol{p}'$ is a satisfying assignment for $\mathsf{x}_{\mathsf{AIR}}$. □

Differences with [BBHR19]. Let us briefly discuss the differences between Construction 2.24 and the IOP protocol of [BBHR19]. In the actual Stark construction, multiple invocations of the RS-IOPP $(\mathcal{P}_{\mathsf{RS}}, \mathcal{V}_{\mathsf{RS}})$ are “batched” into a single one, by testing a random linear combination of the prover's oracles instead of testing them individually. (In [BBHR19, Appendix D], the subprotocol allowing to do this batched verification is called the *algebraic linking IOP* (ALI) protocol). Introduced in [RVW13] in the context of interactive proofs of proximity, this technique adds an additional round of interaction where the verifier sends randomly sampled coefficients to form the linear combination. Proximity tests are expensive subroutines, and this aggregation technique allows to reduce prover and verifier running times. The soundness of this randomized transformation relies on distance preservation lemmas for random linear combinations (Section 1.4.2).

On the technical side, the reduction from multiple instances of the RS proximity problem to a single one requires to take care of the variations between the different prescribed degree bounds. This is done using a standard technique (introduced in [BS08]), which consists in multiplying each oracle by a monomial of well-chosen monomial before forming the random linear combination.

The very same technique enables to adjust any prescribed degree bound to a larger one. Indeed, one of the requirements for using the FRI protocol as a subroutine is that the dimension of the considered Reed-Solomon code is an integral power of two, the other one being that L is a coset of a multiplicative or additive subgroup (see Section 2.1.5).

Moreover, the soundness of the Stark proof system is better than the one stated in Proposition 2.25 and subsequent work further improved the rejection probability of the verifier by sampling outside the domain L (see [BGKS20, Section 5] or [Sta21a, Section 5.3]).

It is worth noting that the construction of [BBHR19] does not require the field to grow with the security parameter. For instance, the first implementation of Stark worked over the field $\mathbb{F}_{2^{64}}$ while being able to reach soundness error $2^{-\kappa}$ for any security parameter κ, by repeating only some parts of the protocol (see [BBHR18b] for more details). Moreover, the Stark construction can operate over a prime field or a binary field, as long as the field admit a large enough additive or multiplicative subgroup of smooth order (in particular larger than the execution time of the computation to be verified).

Finally, Construction 2.24 is not zero-knowledge. The Stark construction [BBHR19] can be made zero-knowledge against query-bounded honest verifier by slightly slackening the degree constraints on $f_1, \ldots, f_w$, which has minimal impact on the efficiency parameters.

CHAPTER 3

Constructing IOPs of Proximity from distance-preserving folding operators

3.1 Generic interactive oracle proof of proximity based on folding operators

From the FRI protocol (Section 2.1.5) from [BBHR18a], we derive a methodology to construct proximity tests for codes in the IOP model from distance-preserving folding operators. From the study of the FRI protocol, this abstraction is quite straightforward to obtain.

Let $\mathbb{F}$ be some finite field. Let us consider an $\mathbb{F}$-linear code $C \subset \Sigma^D$, where Σ is an $\mathbb{F}$-linear space not necessarily equal to $\mathbb{F}$, and D is some evaluation domain.

3.1.1 Folding operators

In this section, we assume that one has defined a finite sequence of codes $(C_i)_{0 \leqslant i \leqslant r}$ for some integer r, where $C_0 \coloneqq C$ and each code $C_i \subset \Sigma^{D_i}$. We will assume that the evaluations domains $(D_i)_{0 \leqslant i \leqslant r}$ satisfy the following. For each $i \in [\![0, r-1]\!]$, assume there exist an integer l_i and a map $\pi_i : D_i \to D_{i+1}$ such that π_i is l_i-to-1 from D_i to $\pi_i(D_i) = D_{i+1}$. In particular, $|D_{i+1}| = \frac{|D_i|}{l_i}$. For any $y \in D_{i+1}$, we will denote $S_y \coloneqq \pi_i^{-1}(\{y\})$ the set of the l_i preimages of y by the function π_i.

Moreover, suppose that for each $i \in [\![0, r-1]\!]$, one can define a family of *folding operators* $\mathbf{Fold}[\cdot, z] : \Sigma^{D_i} \to \Sigma^{D_{i+1}}$ parameterized by $z \in \mathbb{F}^t$ for some positive integer t. These operators are designed to "compress" functions evaluated over D_i into functions over D_{i+1} and feature nice properties with respect to the codes C_i and C_{i+1}.

Definition 3.1 (Folding operator)**.** *A* folding operator *for the code C_i is a map* $\mathbf{Fold}[\cdot, \cdot] : \Sigma^{D_i} \times \mathbb{F}^t \to \Sigma^{D_{i+1}}$ *satisfying the following properties.*

1. *(Completeness) For any $z \in \mathbb{F}^t$,* $\mathbf{Fold}[C_i, z] \subseteq C_{i+1}$.
2. *(Locality) For any function $f : D_i \to \Sigma$, $z \in \mathbb{F}^t$ and $y \in D_{i+1}$, one can compute* $\mathbf{Fold}[f, z](y)$ *by making l_i queries to the function f.*

To ensure soundness of the IOPP based on folding, we will also require that a folding operator preserves the relative distance. Namely, if a function $f : D_i \to \Sigma$ is far from the code C_i, we expect the folding of the function f to be far from the code C_{i+1} with high probability over $z \in \mathbb{F}^t$. We formulate this distance preservation property in terms of relative weighted agreements.

Definition 3.2 (Weighted agreement)**.** *For any weight function $\varphi : D \to [0,1]$, we define the* φ-agreement *of $u, v \in \Sigma^D$, denoted* $\mathsf{agree}_\varphi(u, v)$*, as follows:*

$$\mathsf{agree}_\varphi(u, v) \coloneqq \frac{1}{|D|} \sum_{\substack{x \in D \\ u(x) = v(x)}} \varphi(x).$$

Moreover, given $C \subset \Sigma^D$ and $u \in \Sigma^D$, we define the φ-agreement of u with C, *denoted* $\mathsf{agree}_\varphi(u, C)$, *as*

$$\mathsf{agree}_\varphi(u, C) \coloneqq \max_{v \in C} \mathsf{agree}_\varphi(u, v).$$

Observe that, if a weight function $\varphi : D \to [0,1]$ is constant equal to 1, then agree_φ is the standard notion of relative agreement, *i.e.* for any $u, v \in \Sigma^D$ and any subset $S \in \Sigma^D$, we have

$$\mathsf{agree}_\varphi(u, v) = \frac{1}{|D|} |\{x \in D \mid u(x) = v(x)\}| = 1 - \Delta(u, v),$$

and

$$\mathsf{agree}_\varphi(u, S) = 1 - \Delta(u, S).$$

Consequently, we have the following fact:

Fact 3.3. *For any weight function $\varphi : D \to [0,1]$, any $u, v \in \Sigma^D$ and any $S \subset \Sigma^D$, we have*

$$\mathsf{agree}_\varphi(u, v) \leqslant 1 - \Delta(u, v) \qquad \text{and} \qquad \mathsf{agree}_\varphi(u, S) \leqslant 1 - \Delta(u, S).$$

Definition 3.4 (Distance preservation)**.** *Let us consider a function $\gamma : (0,1) \times [0,1] \to [0,1]$ which is strictly increasing with respect to the second variable. Let $i \in [\![0, r-1]\!]$, and denote by λ_{i+1} the minimum relative distance of C_{i+1}. We say that a folding operator* $\mathbf{Fold}[\cdot, \cdot]$ *satisfies* distance preservation *if, for any weight functions $\varphi_i : D_i \to [0,1]$ and $\varphi_{i+1} : D_{i+1} \to [0,1]$ satisfying*

$$\forall y \in D_{i+1}, \qquad \varphi_{i+1}(y) \geqslant \frac{1}{l_i} \sum_{x \in \pi_i^{-1}(\{y\})} \varphi_i(x), \tag{3.1}$$

any $\varepsilon \in (0,1)$, any $\delta \in (0, \gamma(\varepsilon, \lambda_{i+1}))$ and any function $f : D_i \to \Sigma$ of φ_i-agreement

$$\mathsf{agree}_{\varphi_i}(f, C_i) < 1 - \delta,$$

we have

$$\Pr_{z \in \mathbb{F}^t} \left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}[f, z], C_{i+1}) > 1 - \delta + \varepsilon \right] < \eta,$$

for some $\eta \in (0,1)$.

The reason why we consider weighted agreements instead of the standard relative Hamming distance is that it will facilitate tracking inconsistencies between the oracles actually sent by a malicious prover and the expected prover's messages (prescribed by the protocol) during soundness analysis. The weight functions φ_i, φ_{i+1} are left undefined in Definition 3.4 since weights will be assigned to elements of the supports D_i, D_{i+1} depending on a prover's strategy.

3.1.2 Generic IOPP construction

Now we describe a generic way of constructing a public-coin IOPP to test proximity to a code $C \subseteq \Sigma^D$ using folding operators.

Taking $C_0 = C$ and $D_0 = D$, we consider a sequence of codes $(C_i)_{0 \leqslant i \leqslant r}$ with a family of folding operators defined as per Section 3.1.1. As in the FRI protocol [BBHR18a], our protocol is divided into two phases. The interactive phase is referred to as COMMIT phase, while the non-interactive one is named QUERY phase.

The COMMIT phase is an interaction over r rounds between a prover $\mathscr{P}$ and a verifier $\mathscr{V}$. At each round i, the verifier samples a random element $z_i \in \mathbb{F}^t$. The prover answers with an oracle function $f_{i+1} : D_i \to \Sigma$, expected to be equal to $\textbf{Fold}\left[f_i, z_i\right]$.

During the QUERY phase, the task of the verifier $\mathscr{V}$ is to check that each pair of oracle functions (f_i, f_{i+1}) is consistent. The standard idea is to test whether the equality

$$f_{i+1}(y_{i+1}) = \textbf{Fold}\left[f_i, z_i\right](y_{i+1}) \tag{3.2}$$

holds at a random point $y_{i+1} \in D_{i+1}$. Thanks to the local property of the folding operator, the verifier $\mathscr{V}$ can perform such a test by querying l_i entries of f_i and one entry of f_{i+1}. As in [BBHR18a], we call this step of verification a *round consistency test*. More specifically, the verifier begins by sampling uniformly at random $y_0 \in D_0$ and once this is done, all the locations of the round consistency tests below the current **query test** are determined. Indeed, for each i, $\mathscr{V}$ defines $y_{i+1} \coloneqq \pi_i(y_i)$ to be the point where Equation (3.2) is checked. Through this process, and as in the FRI protocol, the round consistency tests are correlated in order to improve soundness. Such a **query test** can be seen as a *global* consistency test. As a final test, the verifier checks that $f_r \in C_r$ and rejects if it is not the case.

Construction 3.5 (IOPP $(\mathscr{P}, \mathscr{V})$ for a code C based on folding operators)**.**
The prover $\mathscr{P}$ and verifier $\mathscr{V}$ agree on representations of the codes of a given sequence $(C_i)_{0 \leqslant i \leqslant r}$, where $C_0 \coloneqq C$ and each code C_i admits a family of folding operators. The COMMIT *phase is an interaction over r rounds, whereas the* QUERY *phase involves only the verifier.*

Inputs: *The prover receives as explicit input a function $f = f_0 : D_0 \to \Sigma$, expected to be a codeword of C_0. The verifier has oracle access to it.*

COMMIT ***phase:***

1. *For each round i from 0 to $r-1$:*
   (a) ***Verifier*** $\mathscr{V}$ *sends* $z_i \xleftarrow{\$} \mathbb{F}^t$*;*
   (b) ***Prover*** $\mathscr{P}$ *gives oracle access to* $f_{i+1} : D_{i+1} \to \Sigma$ *such that* $f_{i+1} = \textbf{Fold}\left[f_i, z_i\right]$.

QUERY ***phase:***

1. *Repeat α times the following* ***query test****:*
   (a) *Sample $\boldsymbol{y}_0 \in D_0$ uniformly at random;*
   (b) *For $i = 0$ to $r-1$:*
      i. *Define $\boldsymbol{y}_{i+1} \in D_{i+1}$ as $\boldsymbol{y}_{i+1} = \pi_i(y_i)$;*
      ii. *Query l_i entries of f_i to compute $\textbf{Fold}\left[f_i, z_i\right](\boldsymbol{y}_{i+1})$;*
      iii. *Query $f_{i+1}(\boldsymbol{y}_{i+1})$;*
      iv. *If $f_{i+1}(\boldsymbol{y}_{i+1}) \neq \textbf{Fold}\left[f_i, z_i\right](\boldsymbol{y}_{i+1})$, outputs* ***reject*** (Round consistency check) *;*
2. *Outputs acccept if and only if $f_r \in C_r$ (*Final test*).*

On the global consistency check. Suppose that the verifier $\mathscr{V}$ checks whether Equation (3.2) holds at a location y_{i+1} sampled uniformly in D_{i+1} at each round. Denoting p_i the fraction of round consistency checks that pass when testing consistency between f_i and f_{i+1}, the verifier would accepts with probability $\prod_{i=0}^{r-1} p_i$. This could be compensated by increasing the number of random tests, and thus query complexity. For IOPPs with non-constant number of rounds, this would imply a non-constant multiplicative loss in query complexity.

Soundness analysis shows that, a global consistency check has soundness error at most $\prod_{i=0}^{r-1} p_i$ and, depending on the strategy of the prover, this probability can actually be strictly smaller. Let us illustrate this on a simple example.

Example 3.6. Suppose that there are $r = 2$ rounds and $|D_0| = 8, |D_1| = 4, |D_2| = 2$. Now, suppose that the prover sent oracles f_1, f_2 such that $s \in D_1, s' \in D_2$ are the only two values for which f_1 is not consistent with f_0 and f_2 is not consistent with f_1, respectively. Moreover, assume $\pi_1(s) \neq s'$.

Assume that the verifier runs a global consistency test as prescribed. There is only one global consistency check over four that passes, which is the one avoiding the locations $s \in D_1$ and $s' \in D_2$. Then the verifier accepts with probability $\frac{1}{4}$.

Now suppose that, for $i \in [\![0, 1]\!]$, the verifier checks whether Equation (3.2) holds at a location y_{i+1} sampled uniformly at random in D_{i+1}. When testing consistency between f_0 and f_1, the verifier accepts with probability $\frac{3}{4}$, since there is only one location of D_1 that fails the test for this round. Similarly, when testing consistency between f_1 and f_2, the verifier accepts with probability $\frac{1}{2}$. Assuming that the two round consistency checks are independent, the verifier accepts with probability $\frac{3}{8} > \frac{1}{4}$.

Remark 3.7. *Depending on the evaluation codes considered, it may be convenient to adapt the final round as follows in order to avoid the cost of a membership test to C_r. During the last round of the COMMIT phase, instead of sending a codeword $f_r \in C_r$, an honest $\mathscr{P}$ may "unencodes" f_r, meaning he retrieves a word w_r from the messages space of C_r whose encoding leads to $f_r \in C_r$. Prover $\mathscr{P}$ sends k_r message symbols to represent w_r, where k_r refers to the message length of the code C_r. In that case, the verifier no longer needs to run a membership test to the code C_r during the QUERY phase. The verifier $\mathscr{V}$ can re-encode w_r, interpreting f_r to be the the encoding of w_r. This variant of the protocol is the one presented in the FRI protocol [BBHR18a] for Reed-Solomon codes (in that case, w_r is the coefficients of a polynomial of bounded degree).*

Notice that in some cases, e.g. in Construction 2.15, the verifier does not need to encode w_r. Indeed, in the last round of Construction 2.15, the function f_r is expected to be the evaluation of a constant polynomial function.

Theorem 3.8. *Let $(C_i)_{0 \leqslant i \leqslant r}$ be a sequence of codes such that there exists a family of folding operators for each code C_i satisfying Definitions 3.1 and 3.4. The r-rounds IOPP system $(\mathscr{P}, \mathscr{V})$ for the code $C = C_0$ of Construction 3.5 is public-coin and fulfills the following properties:*

Perfect completeness: *If $f \in C$ and if the oracles $f_1, \dots f_r$ are computed by an honest prover $\mathscr{P}$, then $\mathscr{V}$ outputs accept with probability 1.*

Soundness: *Assume $f : D \to \Sigma$ is δ-far from C. For any $\varepsilon \in (0, 1)$ and any unbounded prover $\mathscr{P}^*$, the verifier $\mathscr{V}$ outputs accept after α repetitions of the QUERY phase with probability at most*

$$r\eta + (1 - \min(\delta, \gamma(\varepsilon, \lambda)) + r\varepsilon)^{\alpha},$$

where λ denotes the smallest relative minimum distance of the codes C_i, $i \in [\![0, r]\!]$ and $\gamma(\cdot, \cdot)$ is the function defined in Definition 3.4.

Proof. (Perfect completeness) Assume that $f_0 \in C_0$. An honest prover who follows the prescription of the COMMIT phase will make the round consistency tests pass with probability 1 for all rounds i. By completeness of the folding operator for every round i, we have $f_r \in C_r$. Therefore, the final test also passes. Thus, the verifier always accepts.

(Soundness) Our analysis relies on techniques of proofs from [BGKS20]. A similar analysis appears in [BN20]. We perform our analysis for $\alpha = 1$ repetition of the **query test**. We observe that the soundness error for $\alpha > 1$ directly follows from this case. Let $(f_i)_{1 \leqslant i \leqslant r}$ be the output of the COMMIT phase and $(y_i)_{1 \leqslant i \leqslant r}$ be the query points selected for the QUERY phase. The verifier accepts if both

1. for all $i \in [\![0, r-1]\!]$, $f_{i+1}(y_{i+1}) = \textbf{Fold}\left[f_i, z_i\right](y_{i+1})$,
2. $f_r \in C_r$.

Observe that if $f_r \notin C_r$, the verifier rejects with probability 1, therefore we continue the analysis assuming $f_r \in C_r$. Since the soundness analysis is quite technical, we divide it into several steps to improve readability.

Step 1 – Coloring the graph induced by prover's oracles. Set G the $(r+1)$-layered graph with vertex set $D_0 \sqcup D_1 \sqcup \cdots \sqcup D_r$. The edges of G consist in the couples $(y_i, y_{i+1}) \in D_i \times D_{i+1}$ such that $\pi_i(y_i) = y_{i+1}$. For any edge of G, the vertex y_{i+1} is called the *parent* of y_i. Vertices sharing the same parent are said to be *siblings*. For any vertex within the last layer $y_r \in D_r$, we denote by $G_{|y_r}$ the subgraph of G corresponding to the complete tree with root y_r. Therefore the trees $G_{|y_r}$ are disjoint.

A query test starts by selecting a leaf $y_0 \in D_0$, which belongs to a unique tree $G_{|y_r}$ for a certain $y_r \in D_r$. The verifier queries one set of siblings at each layer $i \in [\![0, r-1]\!]$ of $G_{|y_r}$, whose union forms a subset of vertices of G that we call the *path from* y_0 *to* y_r. Note that a path to y_r does not include y_r.

We now color the vertices of G (except those in the last layer) according to their success in passing the round consistency test. For $i \in [\![0, r-1]\!]$, a vertex $y_i \in D_i$ is colored green if

$$f_{i+1}(\pi_i(y_i)) = \textbf{Fold}\left[f_i, z_i\right](\pi_i(y_i))$$

and colored red otherwise. Notice siblings have the same color. The verifier outputs accept if and only if every vertex along the queried path from y_0 to y_r is green.

Step 2 – Defining the function of weights. Define $\psi_0 : D_0 \to [0,1]$ such that $\psi_0(x) = 1$ if $x \in D_0$ is green and $\psi_0(x) = 0$ otherwise. For all $i \in [1, r-1]$, define function

$$\psi_i : D_i \to [0,1]$$

such that $\psi_i(x)$ is equal to the fraction of leaves $x_0 \in D_0$ for which the path from x_0 to $x \in D_i$ contains only green vertices.

Step 3 – Rejection probability in terms of weighted agreement. By construction, the probability $\mathsf{err}_{\text{query}}$ that the verifier accepts during the QUERY phase is given by

$$\mathsf{err}_{\text{query}} = \frac{1}{|D_r|} \sum_{x \in D_r} \psi_r(x).$$

For $i \in [\![0, r-1]\!]$, let us set

$$\mathsf{agree}_{f_i} \coloneqq \mathsf{agree}_{\psi_i}(f_i, C_i), \tag{3.3}$$

where the ψ-agreement agree_{ψ} is defined in Definition 3.2. Since $f_r \in C_r$, observe that

$$\mathsf{err}_{\text{query}} = \mathsf{agree}_{f_r}. \tag{3.4}$$

Step 4 – Relating agreement of f_i with the one of the folding of f_{i-1}. For $i \in [\![0, r-1]\!]$, we define $E_{i+1} \subseteq D_{i+1}$ to be the set of coordinates where f_{i+1} differs from $\mathbf{Fold}[f_i, z_i]$, i.e.

$$E_{i+1} \coloneqq \left\{ y \in D_{i+1} \mid \forall x \in S_y, x \text{ is red} \right\}.$$

Let us fix $i \in [\![0, r-1]\!]$. We aim to show that

$$\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], C_{i+1}) \geqslant \mathsf{agree}_{\psi_{i+1}}(f_{i+1}, C_{i+1}).$$

Let $v \in C_{i+1}$ such that

$$\mathsf{agree}_{\psi_{i+1}}(f_{i+1}, v) = \mathsf{agree}_{\psi_{i+1}}(f_{i+1}, C_{i+1})$$

(breaking ties arbitrarily). Since for any $y \in E_{i+1}$, $\psi_{i+1}(y) = 0$, we can write

$$\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], v) = \frac{1}{|D_{i+1}|} \sum_{\substack{y \in D_{i+1} \setminus E_{i+1} \\ \mathbf{Fold}[f_i, z_i](y) = v(y)}} \psi_{i+1}(y)$$

and

$$\mathsf{agree}_{\psi_{i+1}}(f_{i+1}, v) = \frac{1}{|D_{i+1}|} \sum_{\substack{y \in D_{i+1} \setminus E_{i+1} \\ f_{i+1}(y) = v(y)}} \psi_{i+1}(y).$$

But $\mathbf{Fold}[f_i, z_i]$ and f_{i+1} coincide on the set $D_{i+1} \setminus E_{i+1}$, hence

$$\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], v) = \mathsf{agree}_{\psi_{i+1}}(f_{i+1}, v).$$

Moreover, we have

$$\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], C_{i+1}) \geqslant \mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], v)$$

by definition of the ψ_{i+1}-agreement. Thus,

$$\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}[f_i, z_i], C_{i+1}) \geqslant \mathsf{agree}_{\psi_{i+1}}(f_{i+1}, C_{i+1}). \tag{3.5}$$

Step 5 – Controlling the weighted agreement after folding. Let $\varepsilon \in (0, 1)$ and

$$\delta_i < \min(1 - \mathsf{agree}_{f_i}, \gamma(\varepsilon, \lambda_i)).$$

Observe that

$$\psi_{i+1}(y) = \begin{cases} 0 & \text{if } y \in E_{i+1}, \\ \dfrac{1}{l_i} \displaystyle\sum_{x \in S_y} \psi_i(x) & \text{if } y \in D_{i+1} \setminus E_{i+1}. \end{cases}$$

Thus, the functions ψ_i satisfy (3.1):

$$\forall y \in D_{i+1}, \psi_{i+1}(y) \geqslant \frac{1}{l_i} \sum_{x \in S_y} \psi_i(x).$$

Since the folding operators satisfy distance preservation (Definition 3.4), we have for all $i \in [\![0, r-1]\!]$

$$\Pr_{z_i \in \mathbb{F}^t}\left[\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}\left[f_i, z_i\right], C_{i+1}) > 1 - \delta_i + \varepsilon\right] \leqslant \eta,$$

which yields

$$\Pr_{z_i \in \mathbb{F}^t}\left[\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}\left[f_i, z_i\right], C_{i+1}) > \max\left(\mathsf{agree}_{f_i}, 1 - \gamma(\varepsilon, \lambda_i)\right) + \varepsilon\right] \leqslant \eta,$$

where agree_{f_i} is the notation introduced in (3.3).

Step 6 – Controlling the weighted agreement of f_r by the one of f_0. Let $\lambda = \min_i(\lambda_i)$. As the function $\gamma(\varepsilon, \cdot)$ is strictly increasing, we have

$$\Pr_{z_i \in \mathbb{F}^t}\left[\mathsf{agree}_{\psi_{i+1}}(\mathbf{Fold}\left[f_i, z_i\right], C_{i+1}) > \max\left(\mathsf{agree}_{f_i}, 1 - \gamma(\varepsilon, \lambda)\right) + \varepsilon\right] \leqslant \eta.$$

Recalling (3.5), we deduce that

$$\Pr_{z_i \in \mathbb{F}^t}\left[\mathsf{agree}_{f_{i+1}} > \max\left(\mathsf{agree}_{f_i}, 1 - \gamma(\varepsilon, \lambda)\right) + \varepsilon\right] \leqslant \eta.$$

By a union bound, the event that for all $i \in [\![0, r-1]\!]$,

$$\mathsf{agree}_{f_{i+1}} \leqslant \max\left(\mathsf{agree}_{f_i}, 1 - \gamma(\varepsilon, \lambda)\right) + \varepsilon$$

occurs with probability at least $1 - r\eta$. If this event occurs, then

$$\mathsf{agree}_{f_r} \leqslant \max\left(\mathsf{agree}_{f_0}, 1 - \gamma(\varepsilon, \lambda)\right) + r\varepsilon.$$

Therefore

$$\Pr_{z_0, \ldots, z_{r-1} \in \mathbb{F}^t}\left[\mathsf{agree}_{f_r} \leqslant \max\left(\mathsf{agree}_{f_0}, 1 - \gamma(\varepsilon, \lambda)\right) + r\varepsilon\right] \geqslant 1 - r\eta.$$

Final step – Putting everything together. Recalling Fact 3.3, we have

$$\mathsf{agree}_{f_0} \leqslant 1 - \Delta(f_0, C_0) < 1 - \delta.$$

Set $\mathsf{err}_{\text{commit}} \coloneqq r\eta$. We deduce that with probability at least $1 - \mathsf{err}_{\text{commit}}$ over the randomness of the verifier during the COMMIT phase, the verifier accepts with probability at most

$$\begin{aligned}\mathsf{err}_{\text{query}} = \mathsf{agree}_{f_r} &\leqslant \max(\mathsf{agree}_{f_0}, 1 - \gamma(\varepsilon, \lambda)) + r\varepsilon \\ &< 1 - \min(\delta, \gamma(\varepsilon, \lambda)) + r\varepsilon.\end{aligned}$$

□

Remark 3.9. *The same proof holds for the variant of the protocol described in Remark 3.7, which results in no change in Theorem 3.8.*

3.2 Distance and correlated agreements with biased sample spaces

In Section 1.4.2, results about distance and correlated agreements of random linear combinations to linear codes were presented. We recall that these results essentially assert that, given a linear code $V \subset \mathbb{F}_q^D$ and a collection of functions $u_0, \ldots, u_{l-1} \in \mathbb{F}_q^D$ such that at least one u_i is δ-far from V, a linear combination $u = \sum_{i=0}^{l-1} r_i u_i$ is also approximately δ-far from V (with high probability over the random choices of $r_0, \ldots, r_{l-1}$).

In the context of proof systems and interactive proximity tests, the coefficients $r_0, \ldots, r_{l-1}$ are each sampled uniformly in $\mathbb{F}_q$ by the verifier. Derandomization techniques can reduce randomness complexity without significantly affecting the soundness error.

A popular choice to reduce randomness complexity (which is used for instance in [BFLS91]) is to sample a single random field element $z \in \mathbb{F}_q$ and set

$$(r_0, r_1, \ldots, r_{l-1}) \coloneqq (1, z, z^2, \ldots, z^{l-1}).$$

More generally, one can reduce the size of the sampling space by setting $r_0, r_1, \ldots, r_{l-1}$ as the output of some small-biased generator G over a finite field[1] $\mathbb{F}_q$. We adopt a definition from [BCL20].

Definition 3.10. *Let $G : \mathbb{F}_q^k \to \mathbb{F}_q^l$ be a function. The function G is ε-biased if*

$$\max_{v \in \mathbb{F}_q^l \setminus \{\mathbf{0}\}} \Pr_{z \in \mathbb{F}_q^k} [\langle v, G(z) \rangle = 0] \leqslant \varepsilon,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product over $\mathbb{F}_q^l$. We define the arithmetic complexity of G as the cost of evaluating G at a point $z \in \mathbb{F}_q^k$.

This section is dedicated to the study of correlated agreements for two simple examples of small-biased generators. They will suffice for the applications to proximity tests for linear codes presented in this manuscript. Relating the bias of the generator with the average distance to linear spaces of linear combinations whose coefficients are output of different classes of small-biased generators remains an intriguing problem.

3.2.1 The case of multilinear combinations

Suppose $l = 2^m$ for some integer m. Let $G \colon \mathbb{F}_q^m \to \mathbb{F}_q^l$ be defined by $G(z) = (z^e)_{e \in \{0,1\}^m}$ for every $z \in \mathbb{F}_q^m$. By applying Lemma 1.19 to a non-zero m-variate polynomial of degree at most m, we get that, for every $v \in \mathbb{F}_q^l \setminus \{\mathbf{0}\}$,

$$\Pr_{z \in \mathbb{F}_q^m} [\langle v, G(z) \rangle = 0] \leqslant \frac{m}{q}.$$

Thus G is $\frac{\log l}{q}$-biased, with arithmetic complexity $O(l)$. This small-biased generator G is used in [BCL20] and will also be used in our IOPP for multivariate polynomial codes in Chapter 4.

[1] Since the influential paper by Naor and Naor [NN90], small-biased generators outputting *bits* have been the most studied. An overview of the numerous applications of small-biased generators in computer science can be found in [Gol08, Section 8.5.2]).

For $\boldsymbol{u} = (u_e)_{e\in\{0,1\}^m} \in \left(\mathbb{F}_q^D\right)^{2^m}$, and $\boldsymbol{z} \in \mathbb{F}_q^m$, we consider the set $H_{\boldsymbol{u}}$ of linear combinations of components of $\boldsymbol{u}$ with coefficients $G(\boldsymbol{z}) \in \left(\mathbb{F}_q\right)^{2^m}$ for some $\boldsymbol{z} \in \mathbb{F}_q^m$, i.e.

$$H_{\boldsymbol{u}} \coloneqq \left\{ \sum_{e\in\{0,1\}^m} \boldsymbol{z}^e u_e \mid \boldsymbol{z} \in \mathbb{F}_q^m \right\}.$$

Given a linear code $V \subset \mathbb{F}_q^D$, the distance of an element u' randomly sampled from $H_{\boldsymbol{u}}$ compared to the maximum distance to V of a member u_e of $\boldsymbol{u}$ is given by the following result.

Proposition 3.11. *Let m be a positive integer. Let $V \subset \mathbb{F}_q^D$ be a linear code of relative distance $\lambda = \Delta(V)$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < 1/3$ and*

$$\delta < 1 - (1 - \lambda + \varepsilon)^{1/3}. \tag{3.6}$$

Let $\boldsymbol{u} = (u_e)_{e\in\{0,1\}^m}$ such that there exists $\boldsymbol{e} \in \{0,1\}^m$ for which $\Delta(u_e, V) > \delta + \varepsilon$. Then

$$\Pr_{z\in\mathbb{F}_q^m}\left[\Delta\left(\sum_{e\in\{0,1\}^m} \boldsymbol{z}^e u_e, V\right) < \delta\right] < \frac{2m}{\varepsilon^2 q}.$$

Proposition 3.11 is the contrapositive of the following proposition, which is in turn based on a result from [BGKS20] (stated in Lemma 1.23). Proposition 3.12 states that if a small number of elements of $S_{\boldsymbol{u}}$ are δ-close to V, then there is a large subset T of coordinates such that the functions $u_{e|T}, \boldsymbol{e} \in \{0,1\}^m$ are codewords of the punctured code $V_{|T}$. The agreement set T being shared by all the functions u_e, this property is called a correlated agreement in [BCI+20].

Proposition 3.12. *Let m be a positive integer. Let $V \subset \mathbb{F}_q^D$ be a linear code of relative distance $\lambda = \Delta(V)$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < 1/3$ and*

$$\delta < 1 - (1 - \lambda + \varepsilon)^{1/3}. \tag{3.7}$$

Let $\boldsymbol{u} = (u_e)_{e\in\{0,1\}^m}$ such that

$$\Pr_{z\in\mathbb{F}_q^m}\left[\Delta\left(\sum_{e\in\{0,1\}^m} \boldsymbol{z}^e u_e, V\right) < \delta\right] \geqslant \frac{2m}{\varepsilon^2 q}. \tag{3.8}$$

Then there exist $T \subset D$ and a family $\boldsymbol{v} = (v_e)_{e\in\{0,1\}^m} \in V^{2^m}$ such that

- $|T| \geqslant (1 - \delta - m\varepsilon)\,|D|$,
- *for each $\boldsymbol{e} \in \{0,1\}^m$, $u_{e|T} = v_{e|T}$.*

Proof. We proceed by induction on the number of variables m. The case $m = 1$ is dealt with in [BGKS20, Lemma 2] (restated in Lemma 1.23). Let us assume that the proposition is true for $m-1$ and prove that it also holds for m. For $\boldsymbol{z} \in \mathbb{F}_q^m$, we write $\boldsymbol{z} = (\boldsymbol{z}', z_m)$, with $\boldsymbol{z}' \in \mathbb{F}_q^{m-1}$ and $z_m \in \mathbb{F}_q$. Similarly, for $\boldsymbol{e} \in \{0,1\}^m$, we write $\boldsymbol{e} = (\boldsymbol{a}, e_m)$, with $\boldsymbol{a} \in \{0,1\}^{m-1}$ and $e_m \in \{0,1\}$. Equation (3.8) gives

$$\Pr_{z_m\in\mathbb{F}_q}\left[\Pr_{\boldsymbol{z}'\in\mathbb{F}_q^{m-1}}\left[\Delta\left(\sum_{\boldsymbol{a}\in\{0,1\}^{m-1}} \boldsymbol{z}'^{\boldsymbol{a}}\left(u_{(\boldsymbol{a},0)} + z_m u_{(\boldsymbol{a},1)}\right), V\right) < \delta\right] \geqslant \frac{2(m-1)}{\varepsilon^2 q}\right] \geqslant \frac{2}{\varepsilon^2 q}.$$

For any $z \in \mathbb{F}_q$, we write $u_{\boldsymbol{a},z} = u_{(\boldsymbol{a},0)} + z u_{(\boldsymbol{a},1)}$. Let A be the set

$$A = \left\{ z \in \mathbb{F}_q ; \Pr_{z' \in \mathbb{F}_q^{m-1}} \left[\Delta \left(\sum_{\boldsymbol{a} \in \{0,1\}^{m-1}} \boldsymbol{z'}^{\boldsymbol{a}} u_{\boldsymbol{a},z}, V \right) < \delta \right] \geqslant \frac{2(m-1)}{\varepsilon^2 q} \right\}.$$

By assumption, $|A| \geqslant 2/\varepsilon^2$. Moreover the inductive hypothesis implies that for each $z \in A$, there exist $T_z \subset D$ and $v_{\boldsymbol{a},z} \in V$ such that

$$|T_z| \geqslant (1 - \delta - (m-1)\varepsilon)\,|D| \text{ and } u_{\boldsymbol{a},z|_{T_z}} = v_{\boldsymbol{a},z|_{T_z}} \quad \text{for all } \boldsymbol{a} \in \{0,1\}^{m-1}.$$

We are now in a position where we can mimic the proof of [BGKS20].

Let us prove there exists a large subset $A' \subset A$ such that for all $\boldsymbol{a} \in \{0,1\}^{m-1}$ and for all $z \in A'$, $v_{\boldsymbol{a},z}$ depends linearly on z, *i.e.* there exists some $v_{(\boldsymbol{a},0)}, v_{(\boldsymbol{a},1)} \in V$ such that $v_{\boldsymbol{a},z} = v_{(\boldsymbol{a},0)} + z v_{(\boldsymbol{a},1)}$.

For z_0, z_1, z_2, picked uniformly and independently in A and y picked uniformly from D, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{z_0,z_1,z_2} \left[\frac{|T_{z_0} \cap T_{z_1} \cap T_{z_2}|}{|D|} \right] &= \mathop{\mathbf{E}}_{y,z_0,z_1,z_2} \left[\mathbf{1}_{y \in T_{z_0} \cap T_{z_1} \cap T_{z_2}} \right] \\
&= \mathop{\mathbf{E}}_{y} \left[\mathop{\mathbf{E}}_{z} \left[\mathbf{1}_{y \in T_z} \right]^3 \right] \\
&\geqslant \mathop{\mathbf{E}}_{y,z} \left[\mathbf{1}_{y \in T_z} \right]^3 \\
&\geqslant (1-\delta)^3 \\
&\geqslant 1 - \lambda + \varepsilon.
\end{aligned}$$

From this, one obtains:

$$\Pr_{z_0,z_1,z_2} \left[|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1-\lambda)\,|D| \right] \geqslant \varepsilon.$$

The probability of z_0, z_1, z_2 being all distinct is at least $1 - \frac{3}{|A|}$, which is greater than $1 - \frac{\varepsilon}{2}$ since $|A| \geqslant \frac{2}{\varepsilon^2} > \frac{6}{\varepsilon}$. Thus, we get

$$\Pr_{z_0,z_1,z_2} \left[z_0, z_1, z_2 \text{ are all distinct and } |T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1-\lambda)\,|D| \right] \geqslant \varepsilon/2.$$

Consequently, there are distinct z_1 and z_2 such that

$$\Pr_{z_0} \left[|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1-\lambda)\,|D| \right] \geqslant \varepsilon/2.$$

Fix $z_0 \in \mathbb{F}_q$ such that $|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1-\lambda)\,|D|$ and set $S \coloneqq T_{z_0} \cap T_{z_1} \cap T_{z_2}$. For each $\boldsymbol{a} \in \{0,1\}^{m-1}$, the vectors

$$\left(z_0, u_{\boldsymbol{a},z_0}\right), \left(z_1, u_{\boldsymbol{a},z_1}\right), \left(z_2, u_{\boldsymbol{a},z_2}\right)$$

are collinear. Then their restrictions to S, $\left(z_i, u_{\boldsymbol{a},z_i|_S}\right)$, which coincide with $\left(z_i, v_{\boldsymbol{a},z_i|_S}\right)$ by definition of S, are also collinear. Since $|S| \geqslant (1-\lambda)\,|D|$ and λ is the minimum distance of V, we can linearly map the vectors $v_{\boldsymbol{a},z_i|_S}$ to elements $v_{\boldsymbol{a},z_i}$ of the code V, which preserves collinearity. Therefore, the vectors $v_{\boldsymbol{a},z_i}$ ($z = z_0, z_1, z_2$) all belong to a same line

$$\left\{ v_{(\boldsymbol{a},0)} + z v_{(\boldsymbol{a},1)} ; z \in \mathbb{F}_q \right\} \subset \mathbb{F}_q^D,$$

where $v_{(\boldsymbol{a},0)}, v_{(\boldsymbol{a},1)} \in V$. Set $A' = \left\{ z \in A \mid v_{\boldsymbol{a},z} = v_{(\tilde{\boldsymbol{e}},0)} + z v_{(\boldsymbol{a},1)} \right\}$. Then we have $|A'| \geqslant \frac{\varepsilon}{2} |A| \geqslant \frac{1}{\varepsilon}$. Now consider the set

$$T = \left\{ x \in D | \forall \boldsymbol{a} \in \{0,1\}^{m-1},\ u_{(\boldsymbol{a},0)}(x) = v_{(\boldsymbol{a},0)}(x) \text{ and } u_{(\boldsymbol{a},1)}(x) = v_{(\boldsymbol{a},1)}(x) \right\}.$$

For any $x \in D \backslash T$, there exists at most one element $z \in \mathbb{F}_q$ such that, for all $\boldsymbol{a} \in \{0,1\}^{m-1}$,

$$u_{(\boldsymbol{a},0)}(x) + z u_{(\boldsymbol{a},1)}(x) = v_{(\tilde{\boldsymbol{e}},0)}(x) + z v_{(\boldsymbol{a},1)}(x).$$

For any $z \in A'$, for any $\boldsymbol{a} \in \{0,1\}^{m-1}$, we have

$$1 - \frac{|T_z|}{|D|} \geqslant \Delta(u_{\boldsymbol{a},z}, v_{\boldsymbol{a},z}).$$

We thus also have

$$\begin{aligned} 1 - \frac{|T_z|}{|D|} &\geqslant \mathop{\mathbf{E}}_{z \in A'} \left[\Delta(u_{\boldsymbol{a},z}, v_{\boldsymbol{a},z}) \right] \\ &\geqslant \frac{|D \backslash T|}{|D|} \left(1 - \frac{1}{|A'|} \right) \\ &\geqslant \left(1 - \frac{|T|}{|D|} \right) (1 - \varepsilon) \\ &\geqslant 1 - \frac{|T|}{|D|} - \varepsilon \end{aligned}$$

Using $|T_z| \geqslant (1 - \delta - (m-1)\varepsilon) |D|$, and rearranging, we get $|T| \geqslant (1 - \delta - m\varepsilon) |D|$. □

As suggested in Section 3.1, applications to proximity tests may require a variant of Proposition 3.12 stated in terms of weighted agreements.

Proposition 3.13. *Let m be a positive integer. Let $V \subset \mathbb{F}_q^D$ be a linear code of distance $\lambda = \Delta(V)$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < 1/3$ and*

$$\delta < 1 - (1 - \lambda + \varepsilon)^{1/3}.$$

For any weight function $\varphi : D \to [0,1]$ and any $\boldsymbol{u} = (u_{\boldsymbol{e}})_{\boldsymbol{e} \in \{0,1\}^m}$ satisfying

$$\Pr_{z \in \mathbb{F}_q^m} \left[\mathsf{agree}_\varphi \left(\sum_{\boldsymbol{e} \in \{0,1\}^m} z^{\boldsymbol{e}} u_{\boldsymbol{e}}, V \right) > 1 - \delta \right] \geqslant \frac{2m}{\varepsilon^2 q}, \tag{3.9}$$

there exist $T \subset D$ and a family $\boldsymbol{v} = (v_{\boldsymbol{e}})_{\boldsymbol{e} \in \{0,1\}^m} \in V^{2^m}$ such that

- $\sum_{x \in T} \varphi(x) \geqslant (1 - \delta - m\varepsilon) |D|$,
- *for each $\boldsymbol{e} \in \{0,1\}^m$, $u_{\boldsymbol{e}|T} = v_{\boldsymbol{e}|T}$.*

Before proving Proposition 3.13, we first state a variant of Lemma 1.23 due to [BGKS20]. The proof of Lemma 3.14 is relatively straigthforward, based on the original proof of [BGKS20, Lemma 3.2].

Lemma 3.14 ([BGKS20, Lemma 2] – weighted agreement version)**.** *Let $V \subset \mathbb{F}_q^D$ be a linear code of distance $\lambda = \Delta(V)$. Let $\varepsilon, \delta > 0$ such that $\varepsilon < 1/3$ and*

$$\delta < 1 - (1 - \lambda + \varepsilon)^{1/3}.$$

For any weight function $\varphi : D \to [0,1]$ and any functions $u_0, u_1 \in \mathbb{F}_q^D$ satisfying

$$\Pr_{z \in \mathbb{F}_q}\left[\mathsf{agree}_\varphi(u_0 + zu_1, V) > 1 - \delta\right] \geqslant \frac{2}{\varepsilon^2 q}, \tag{3.10}$$

there exist $T \subset D$ and $v_0, v_1 \in V$, such that

- $\sum_{x \in T} \varphi(x) \geqslant (1 - \delta - \varepsilon)\,|D|$,
- *for each $i \in \{0,1\}$, $u_{i|T} = v_{i|T}$.*

Proof. By Fact 3.3, the set

$$\left\{z \in \mathbb{F}_q \mid \mathsf{agree}_\varphi(u_0 + zu_1, V) > 1 - \delta\right\}$$

is contained in the set

$$A \coloneqq \left\{z \in \mathbb{F}_q \mid \Delta(u_0 + zu_1, V) < \delta\right\}$$

of size $|A| \geqslant \frac{2}{\varepsilon^2}$ (by assumption). Now, the proof follows the one of [BGKS20, Lemma 2].

For each $z \in \mathbb{F}_q$, denote $u_z = u_0 + zu_1$ and let $v_z \in V$ be a codeword such that

$$\Delta(u_z, V) = \Delta(u_z, v_z).$$

Let $T_z \coloneqq \{x \in D \mid u_z(x) = v_z(x)\}$ be the agreement set of u_z and v_z. For z_0, z_1, z_2, picked uniformly and independently in A and y picked uniformly from D, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{z_0,z_1,z_2}\left[\frac{|T_{z_0} \cap T_{z_1} \cap T_{z_2}|}{|D|}\right] &= \mathop{\mathbf{E}}_{y,z_0,z_1,z_2}\left[\mathbf{1}_{y \in T_{z_0} \cap T_{z_1} \cap T_{z_2}}\right] \\
&= \mathop{\mathbf{E}}_{y}\left[\mathop{\mathbf{E}}_{z}\left[\mathbf{1}_{y \in T_z}\right]^3\right] \\
&\geqslant \mathop{\mathbf{E}}_{y,z}\left[\mathbf{1}_{y \in T_z}\right]^3 \\
&\geqslant (1 - \delta)^3 \\
&\geqslant 1 - \lambda + \varepsilon.
\end{aligned}$$

From this, one obtains

$$\Pr_{z_0,z_1,z_2}\left[|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1 - \lambda)\,|D|\right] \geqslant \varepsilon.$$

The probability of z_0, z_1, z_2 being all distinct is at least $1 - \frac{3}{|A|}$, which is greater than $1 - \frac{\varepsilon}{2}$ since $|A| > \frac{6}{\varepsilon}$. Thus, we get

$$\Pr_{z_0,z_1,z_2}\left[z_0, z_1, z_2 \text{ are all distinct and } |T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1 - \lambda)\,|D|\right] \geqslant \varepsilon/2.$$

Consequently, there are distinct z_1 and z_2 such that

$$\Pr_{z_0}\left[|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1 - \lambda)\,|D|\right] \geqslant \varepsilon/2.$$

Fix a z_0 such that

$$|T_{z_0} \cap T_{z_1} \cap T_{z_2}| \geqslant (1-\lambda)\,|D|\,,$$

and let $S = T_{z_0} \cap T_{z_1} \cap T_{z_2}$. We have that $u_{z_0}, u_{z_1}, u_{z_2}$ all lie on the line

$$l \coloneqq \left\{u_0 + zu_1 : z \in \mathbb{F}_q\right\} \subset \mathbb{F}_q^D.$$

As a consequence, when restricted to S, we have that $u_{z_0|S}, u_{z_1|S}, u_{z_2|S}$ all lie on the line

$$l_{|S} = \left\{u_{0|S} + zu_{0|S} : z \in \mathbb{F}_q\right\} \subset \mathbb{F}_q^S.$$

By definition of S, T_{z_0}, T_{z_1} and T_{z_2}, we also have that $v_{z_0|S}, v_{z_1|S}, v_{z_2|S}$ lie on the line $l_{|S}$. Since $|S| \geqslant (1-\lambda)\,|D|$ and λ is the minimum distance of V, we can linearly reencode $v_{z_0|S}, v_{z_1|S}, v_{z_2|S}$ into $v_{z_0}, v_{z_1}, v_{z_2}$, and we observe that v_{z_0}, v_{z_1} and v_{z_2} all lie on a same line. Thus, there are $v_0, v_1 \in \mathbb{F}_q^D$ such that this line is defined by

$$\left\{v_0 + zv_1; z \in \mathbb{F}_q\right\} \subset \mathbb{F}_q^D.$$

There are $\frac{\varepsilon}{2}$-fraction of the $z_0 \in A$ such that v_{z_0} belongs to this line. Note that for such z_0,

$$v_{z_0} = v_0 + z_0 v_1.$$

Let $A' \subset A$ be the set of elements z's such that v_z (the word closest to u_z) can be written

$$v_z = v_0 + zv_1.$$

Then, we have $|A'| \geqslant \frac{\varepsilon}{2}\,|A| \geqslant \frac{1}{\varepsilon}$ and for all $z \in A'$,

$$\mathsf{agree}_\varphi(u_0 + zu_1, v_0 + zv_1) > 1 - \delta.$$

Therefore,

$$\begin{aligned}
1 - \delta &< \frac{1}{|A'|} \sum_{z \in A'} \mathsf{agree}_\varphi(u_z, v_z) \\
&< \frac{1}{|A'|\,|D|} \sum_{z \in A'} \sum_{x \in D} \left(\varphi(x) \cdot \mathbf{1}_{u_z(x) = v_z(x)}\right) \\
&< \frac{1}{|D|} \sum_{x \in D} \varphi(x) \cdot \left(\frac{1}{|A'|} \sum_{z \in A'} \mathbf{1}_{u_z(x) = v_z(x)}\right).
\end{aligned}$$

Let us consider

$$T \coloneqq \{x \in D \mid u_0(x) = v_0(x) \text{ and } u_1(x) = v_1(x)\}\,.$$

Given $x \in D \backslash T$, there is at most one element $z \in \mathbb{F}_q$ such that

$$u_0(x) + zu_1(x) = v_0(x) + zv_1(x).$$

Thus, we conclude that

$$\begin{aligned}
1 - \delta &< \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \frac{1}{|D|} \sum_{x \in D \backslash T} \varphi(x) \frac{1}{|A'|} \\
&< \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \varepsilon.
\end{aligned}$$

□

Proof of Proposition 3.13. As for Proposition 3.12, we proceed by induction on m. The case $m = 1$ is treated by Lemma 3.14. Let us assume that the statement is true for $m-1$. Recalling Fact 3.3, we have

$$\left\{ z \in \mathbb{F}_q^m \mid \mathsf{agree}_\varphi \left(\sum_{e \in \{0,1\}^m} z^e u_e, V \right) > 1 - \delta \right\} \subseteq \left\{ z \in \mathbb{F}_q^m \mid \Delta \left(\sum_{e \in \{0,1\}^m} p^e u_e, V \right) < \delta \right\}.$$

It follows from (3.9) that the largest set has size at least $\frac{2m}{\varepsilon^2} q^{m-1}$. Then, the proof follows the proof of Proposition 3.12, until we get a set $A' \subset A$ of size at least $1/\varepsilon$ and $v_{(a,0)}, v_{(a,1)} \in V$ such that for all $a \in \{0,1\}^{m-1}$, for all $z \in A'$,

$$v_{a,z} = v_{(a,0)} + z v_{(a,1)}.$$

Let T be the set

$$T \coloneqq \left\{ x \in D | \text{ for all } a \in \{0,1\}^{m-1},\ u_{(a,0)}(x) = v_{(a,0)}(x) \text{ and } u_{(a,1)}(x) = v_{(a,1)}(x) \right\}.$$

For all $a \in \{0,1\}^{m-1}$, for all $z \in A'$,

$$\Delta \left(u_{(a,0)} + z u_{(a,1)}, v_{(a,0)} + z v_{(a,1)} \right) < \delta + (m-1)\varepsilon.$$

Denoting

$$u_{a,z} \coloneqq u_{(a,0)} + z u_{(a,1)} \quad \text{and} \quad v_{a,z} \coloneqq v_{(a,0)} + z v_{(a,1)},$$

we get

$$\mathsf{agree}_\varphi(u_{a,z}, v_{a,z}) > 1 - \delta - (m-1)\varepsilon.$$

Averaging, we have:

$$\begin{aligned} 1 - \delta - (m-1)\varepsilon &< \frac{1}{|A'|} \sum_{z \in A'} \mathsf{agree}_\varphi(u_{a,z}, v_{a,z}) \\ &< \frac{1}{|A'|\,|D|} \sum_{z \in A'} \sum_{x \in D} \left(\varphi(x) \cdot \mathbf{1}_{u_{a,z}(x) = v_{a,z}(x)} \right) \\ &< \frac{1}{|D|} \sum_{x \in D} \varphi(x) \cdot \left(\frac{1}{|A'|} \sum_{z \in A'} \mathbf{1}_{u_{a,z}(x) = v_{a,z}(x)} \right). \end{aligned}$$

For $x \in D \backslash T$, there is at most one element $z \in \mathbb{F}_q$ such that

$$u_{(a,0)}(x) + z u_{(a,1)}(x) = v_{(a,0)}(x) + z v_{(a,1)}(x).$$

Thus, we get

$$\begin{aligned} 1 - \delta - (m-1)\varepsilon &< \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \frac{1}{|D|} \sum_{x \in D \backslash T} \varphi(x) \frac{1}{|A'|} \\ &< \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \varepsilon. \end{aligned}$$

Rearranging, we have $\sum_{x \in T} \varphi(x) > (1 - \delta - m\varepsilon)\,|D|$. □

3.2.2 The case of low-degree parametrized curves

We conclude this section with a small-biased generator commonly used in constructions of proof systems (e.g. [BFLS91]) Let us consider $G\colon \mathbb{F}_q \to \mathbb{F}_q^l$ defined, for every $z \in \mathbb{F}_q$, by

$$G(z) = (1, z, z^2, \dots, z^{l-1}).$$

For every $\boldsymbol{v} \in \mathbb{F}_q^l \setminus \{\mathbf{0}\}$, the polynomial $\sum_{i=0}^{l-1} v_i X^i \in \mathbb{F}_q[X]$ has at most $l-1$ roots. Thus

$$\Pr_{z\in\mathbb{F}_q}[\langle \boldsymbol{v}, G(z)\rangle = 0] \leqslant \frac{l-1}{|\mathbb{F}_q|},$$

i.e. G is $\frac{l-1}{q}$-biased. The state-of-the-art result regarding correlated agreements for generic linear code C and low-degree parametrized curve is due to [BKS18].

Notation 1. *For $\varepsilon \in (0,1)$, denote $J_\varepsilon : [0,1] \to [0,1]$ the function*

$$J_\varepsilon\colon x \mapsto 1 - \sqrt{1 - x(1-\varepsilon)}.$$

Moreover, we denote by J_ε^l the function $J_\varepsilon^l \coloneqq \underbrace{J_\varepsilon \circ J_\varepsilon \circ \cdots \circ J_\varepsilon}_{l \text{ times}}$.

Theorem 3.15 ([BKS18, Theorem 4.5])**.** *Let $V \subset \mathbb{F}_q^D$ be a linear code of relative distance $\lambda = \Delta(V)$. Let $\varepsilon, \delta > 0$ such that and $\delta < J_\varepsilon^l(\lambda)$. Let $u_0, \dots, u_{l-1} \in \mathbb{F}_q^D$ satisfying*

$$\Pr_{z\in\mathbb{F}_q}\left[\Delta\left(\sum_{i=0}^{l-1} z^i u_i, V\right) < \delta\right] \geqslant \frac{l-1}{|\mathbb{F}_q|}\left(\frac{2}{\varepsilon}\right)^{l+1}. \tag{3.11}$$

Then, there exist $T \subset D$, and $v_0, \dots, v_{l-1} \in V$ such that:

- $|T| \geqslant (1-\delta-\varepsilon)|D|$
- *for each $i \in [\![0, l-1]\!]$, $u_{i|T} = v_{i|T}$.*

For our applications, we will need a variant of [BKS18, Theorem 4.5] in terms of weighted agreements. It can be proved by adapting of the proof of [BKS18, Theorem 4.5], and we only highlight the changes to be made.

Proposition 3.16. *Let $\varphi \in [0,1]^D$ be a weight function and $\varepsilon, \delta > 0$ such that and $\delta < J_\varepsilon^l(\lambda)$. Let $u_0, \dots, u_{l-1} \in \mathbb{F}_q^D$ such that*

$$\Pr_{z\in\mathbb{F}_q}\left[\mathsf{agree}_\varphi\left(\sum_{i=0}^{l-1} z^i u_i, V\right) > 1-\delta\right] \geqslant \frac{l-1}{|\mathbb{F}_q|}\left(\frac{2}{\varepsilon}\right)^{l+1}, \tag{3.12}$$

then there exists $T \subset D$, and $v_0, \dots, v_{l-1} \in V$ such that:

- $\sum_{x\in T} \varphi(x) \geqslant (1-\delta-\varepsilon)|D|$
- *for each $i \in [\![0, l-1]\!]$, $u_{i|T} = v_{i|T}$.*

Proof. We only need to prove that the codewords $v_0, \dots, v_{l-1} \in V$ exhibited in Theorem 3.15 satisfy the first item of Proposition 3.16. For $z \in \mathbb{F}_q$ and $(v_0, \dots, v_{l-1}) \in V^l$, let us set

$$v_z \coloneqq \sum_{i=0}^{l-1} z^i v_i.$$

Rewriting the proof of [BKS18, Theorem 4.5] with

$$A = \left\{ z \in \mathbb{F}_q \mid \mathsf{agree}_\varphi\left(u_z, V\right) > 1 - \delta \right\}$$

provides $v_0, \dots, v_{l-1} \in V$ and a set

$$C \coloneqq \left\{ z \in \mathbb{F}_q \mid \mathsf{agree}_\varphi\left(u_z, v_z\right) > 1 - \delta \right\} \subset A$$

with cardinality $|C| > \frac{l-1}{\varepsilon}$. Let us consider the set

$$T \coloneqq \{x \in D \mid u_{i|T} = v_{i|T} \text{ for all } i \in [\![0, l-1]\!]\}.$$

We have:

$$\begin{aligned} 1 - \delta &< \frac{1}{|C|} \sum_{z \in C} \mathsf{agree}_\varphi\left(u_z, v_z\right) \\ &= \frac{1}{|C| \cdot |D|} \sum_{z \in C} \sum_{x \in D} \varphi(x) \mathbb{1}_{u_z(x) = v_z(x)} \\ &= \frac{1}{|D|} \sum_{x \in D} \varphi(x) \frac{1}{|C|} \sum_{z \in C} \mathbb{1}_{u_z(x) = v_z(x)}. \end{aligned}$$

Notice that if there exists $i \in [\![0, l-1]\!]$ such that u_i which does not coincide with v_i, the number of $z \in \mathbb{F}_q$ such that $u_z(x) = v_z(x)$ is at most $l - 1$. Then

$$\begin{aligned} 1 - \delta &\leqslant \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \frac{1}{|D|} \sum_{x \in C \setminus T} \varphi(x) \frac{l-1}{|C|} \\ &\leqslant \frac{1}{|D|} \sum_{x \in T} \varphi(x) + \varepsilon, \end{aligned}$$

which gives the first item of the proposition. □

Chapter 4

Proximity testing for multivariate polynomial codes

The focus of this chapter is to tackle the low-degree testing problem for an oracle function $f : L^m \to \mathbb{F}_q$ and a degree $d < |L|$ in the IOP model. Depending on whether d is a bound on the total degree bound or the individual degrees, this problem can be solved by constructing an IOPP for Reed-Muller codes or tensor product of Reed-Solomon codes. Note that the interesting regime for applications to constructions of proof systems is when m is considered to be a constant, since in that case, multivariate polynomial codes can have constant rate and constant distance when $|L|$ grows.

Our IOPPs for multivariate polynomial codes are generalizations of the FRI protocol to the multivariate case. As for the FRI protocol (Section 2.1.5), we focus on codes whose alphabet $\mathbb{F}_q$ admits either multiplicative or additive subgroups of large 2-smooth order. According to Section 3.1.2, constructing such IOPPs boils down to defining distance-preserving folding operators on a suitable sequence of codes.

While multivariate polynomial codes are locally testable, local testers have query complexity which is linear in d. We construct IOPPs for some families of products of Reed-Solomon codes and Reed-Muller codes with linear prover running time and proof length, and logarithmic verification and query complexity (recall that we count in field operations and give complexities with respect to the length of the code). Since our constructions are explicit, all efficiency measures of the two IOPPs are explicitly presented. These parameters match the IOPP for Reed-Solomon codes of [BBHR18a], from which they are inspired.

On testing proximity to products of codes. Tensor codes are robust locally testable codes [BS06, Vid15, CMS17]. One possible approach to construct IOPP for tensor codes is to rely on a random axis-parallel test. Suppose $C \subset \mathbb{F}^n$ is a linear code and $m \geqslant 3$. A natural local test to check proximity of a function f to the tensor code $C^{\otimes m}$ is to sample $b \in [n], j \in [m]$, query the purported word f on the plane

$$P_{b,j} \coloneqq \left\{(x_1, \ldots, x_m) \mid x_j = b \text{ and, for } i \neq j, x_i \in [n]\right\},$$

then check whether the restriction of f to $P_{b,j}$ is a codeword of $C^{\otimes m-1}$. This gives a local test with query complexity sublinear in the length of $C^{\otimes m}$. Viderman [Vid15], building on the work of [BS06], showed that for $m \geqslant 3$, the local test mentioned above for $C^{\otimes m}$ is also $\frac{\Delta(C)^m}{12}$-robust, where $\Delta(C)$ is the relative minimum distance of C. Combining the robust local tester of [BS06, Vid15] with [Mie09]'s PCPP, [BCG+17] proposed a 1-round IOPP for tensor codes $C^{\otimes m}$ with non-trivial query complexity for $m \geqslant 3$. However, this approach gives a rather poor soundness error, namely smaller than $1 - \frac{\Delta(C)^{O(m)}}{\mathsf{poly}(m)}\delta$. Our aim is to construct proximity tests with, in particular, a better trade-off between query complexity and soundness error, and a folding approach allows us to achieve it.

For starters, let us briefly describe a natural "folding approach" for tensor product codes that would not allow to achieve logarithmic verification. Let $C \subset \mathbb{F}_q^{[n]}$ be an arbitrary linear code, and suppose that we want to test proximity of a function $f : [n]^m \to \mathbb{F}_q$ to the tensor code $C^{\otimes m}$. Considering the sequence of codes $(C_i)_{0 \leqslant i < m}$ where $C_i \coloneqq C^{\otimes m-i}$, there is a simple family of folding operators for each code C_i.

Given a function $f : [n]^{m-i} \to \mathbb{F}_q$ and $z \in \mathbb{F}_q^n$, define $\mathbf{Fold}[f, z] : [n]^{m-i-1} \to \mathbb{F}_q$ by setting, for all $x \in [n]^{m-i-1}$,

$$\mathbf{Fold}[f, z](x) = \sum_{j \in [n]} z_j f(j, x).$$

Such folding operators satisfy Definition 3.1. Indeed, by definition of tensor product of codes and linearity, we have that $f \in C^{\otimes m-i}$ implies that for any $z \in \mathbb{F}_q^n$, $\mathbf{Fold}[f, z] \in C^{\otimes m-i-1}$. Moreover, for any z, $\mathbf{Fold}[f, z](x)$ can be computed by making n queries to f. The queried locations correspond to the preimages of x by the map $\pi_i : [n]^{m-i} \to [n]^{m-i-1}$ which sends $(x_1, \ldots, x_{m-i})$ onto $(x_2, \ldots, x_{m-i})$. By arguing that such folding operators also satisfy Definition 3.4 (for instance, by relying on Corollary 1.24), then an IOPP could be obtained from the framework presented in Section 3.1.2.

However, such an approach would lead to a $(m-1)$-round IOPP for $C^{\otimes m}$, with proof length $< n^m$ (from a geometric sum), query complexity mn, prover complexity $O(n^m)$ and verifier complexity $O(mn) + O(n^2)$ (the first term correponds to the round consistency checks, while the second term is the cost of a membership test for C). In particular, verification time would be sublinear but superlogarithmic in n^m for $m > 3$.

In the specific case where C is a Reed-Solomon code, we will show that one can achieve exponential savings regarding verification. We can indeed construct an IOPP with $O(m \log n)$ queries and soundness error approximately $1 - \delta$ (for suitable parameters) for any values of m. Verifier will also be strictly logarithmic in the length of the tensor code.

4.1 Related work

Tensor product of Reed-Solomon codes (individual degree tests). Low-degree tests for bounded individual degree appear in numerous constructions of probabilistic proof systems [BFL90, BFLS91, PS94, FHS94, ALM+98, RS97, FGL+96, BS08] and play a central role in constructing short PCPs [PS94, BS08, Mie09]. The common idea of such tests is to rely on the following characterization. A function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ is a m-variate polynomial function of individual degrees at most d if and only if, for any k-dimensional axis-parallel affine subspace S of $\mathbb{F}_q^m$, the restriction of f to S is a k-variate polynomial of individual degree d.

Ben-Sasson and Sudan [BS08] constructed a PCPP for the tensor product of RS codes by relying on their PCPP for Reed-Solomon codes. The PCPP to test a function $f : L^m \to \mathbb{F}$ is composed by a PCPP for Reed-Solomon codes (RS-PCPP) for each restrictions of f to an axis-parallel line. Therefore, the prover needs to compute $m|L|^{m-1}$ RS-PCPP, which yields prover complexity and proof length less than $m|L|^m \log^{1.5} |L|$. Both verifier complexity and query complexity are polylogarithmic in $|L|$. Our IOPP for the tensor of RS codes outperforms on all these parameters.

In the IOP model, there is no IOPP specifically tailored for tensor product of Reed-Solomon codes.

Ron-Zewi and Rothblum [RR20] proposed an IOPP for any language computable in deterministic polynomial time and bounded space. In particular, their result gives an IOPP with linear proof

Scheme	Prover	Verifier	Query	Length	Rounds
[BS08, BCGT13]	$O(mn^m \log^{1.5} n)$	$\text{polylog}(n)$	$\text{polylog}(n)$	$O(mn^m \log^{1.5} n)$	0.5
[BCG+17]*	$o(n^m)$	$\text{poly}(m + \log n)$	$O(1)$	$o(n^m)$	1
[RR20]	$\text{poly}(n^m)$	$(n^m)^\varepsilon$	$O(1)$	$< n^m$	$O(1)$
[BCG20]	$O(mn^m \log n)$	$O(nm \log n)$	$O(nm)$	$O(n^m)$	m
This work	$< 8n^m$	$< 8 \log n^m$	$< \log n^m$	$< n^m$	$< \log n^m$

*: restricted to $m \geqslant 3$ and proximity parameter δ smaller than half the minimum distance of the tensor code.

Figure 4.1: Partial comparison of proofs of proximity for tensor product of RS codes of length n^m. Soundness is omitted since it is difficult to provide and compare uniformly.

length and constant query complexity for Reed-Muller codes and tensor product of Reed-Solomon codes. However, the construction proposed by [RR20] has polynomial prover complexity and sublinear verifier complexity and relies on the generic PCP of Proximity for nondeterministic languages of [Mie09], which is commonly believed to be impractical. In contrast, we propose IOP of Proximity for multivariate polynomial codes with a linear-time prover and logarithmic-time verifier. The algorithms of the prover and the verifier have not only better asymptotic complexities, but are also straightforward to implement in practice. Indeed, the tasks of our prover and verifier are nothing more than univariate polynomial interpolations (from a fixed constant number of evaluation points).

Although not specifically designed for multivariate polynomial codes, there are a couple of IOPP constructions for m-wise tensor product of a generic linear code C. Indeed, axis-parallel tests enable local testability of repeated tensor products of any linear codes [BS06, Vid15, CMS17]. Ben-Sasson *et al.* [BCG+17] suggested a 1-round IOPP system for tensor product codes $C^{\otimes m}$, where C is an arbitrary linear code and $m \geqslant 3$. Through interactive proof composition, Ben-Sasson *et al.* combine the robust local tester of [BS06, Vid15, CMS17] for tensor product codes with the Mie's PCP of Proximity for non-deterministic languages [Mie09]. The IOPP system constructed there has sublinear proof length and constant query complexity, which is significantly better than our protocol. However, for fixed $m > 3$, the verifier in [BCG+17] runs in time which is polylogarithmic in the length n of the base code C, whereas our verifier decision complexity is strictly logarithmic in n. Besides, and as opposed as our work, the IOPP system of [BCG+17] requires the proximity parameter to be smaller than half the minimum distance of the tensor code. Our construction is arguably much simpler to implement, as we do not rely on an heavy PCPP for NTIME, like Mie's one [Mie09].

Recently, Bootle, Chiesa and Groth [BCG20] showed how to construct a m-rounds IOPP for tensor codes $C^{\otimes m}$, where C is an arbitrary linear code of length n and dimension k. Their construction also relies on a folding operation (inspired by the FRI protocol of [BBHR19]) but takes a different approach than ours due to their need to work with linear-time encodable codes. In particular, performing the folding operation defined in [BCG20] requires to run an encoding algorithm for the m-wise tensor code $C^{\otimes m}$. When considering C a Reed-Solomon code, best known encoding algorithms run in time at least quasi-linear in n. In contrast, our IOPP does not rely on any encoding procedure of neither the tensor code, nor the base code.

Reed-Muller codes (total degree tests). A substantial body of research studies low total degree test [GLR+91, RS92, RS96, RS97, AS03, BSVW03, MR08] with evaluations over the entire domain $\mathbb{F}_q^m$. For this setting, considering restrictions of f to affine subspaces of fixed dimension is quite

natural. Indeed, if $f : \mathbb{F}_q^m \to \mathbb{F}_q$ has total degree at most d then all its restrictions to u-dimensional affine subspaces are u-variate polynomials of degree at most d. Note that since those affine subspace are not "parallel to the axis", local testers for Reed-Muller codes can have a better soundness error than the axis-parallel tests for tensor product of codes.

For example, the "line-versus-point" test of Rubinfeld and Sudan [RS96] consists in checking the restriction of the function f to a randomly chosen line in $\mathbb{F}_q^m$. Analyses [RS96, AS03, ALM+98] showed that if the test accepts a function f with probability δ, then f agrees with a degree-d polynomial on $\simeq \delta$ fraction of points. The verifier queries $O(d^3)$ field elements to achieve constant soundness error. The original low-degree test of [RS96] can be reformulated in terms of a PCPP if we consider that an auxiliary oracle is given in addition to f. Such oracle proof is supposed to contain the restrictions of f to every line, represented as the $d+1$ coefficients of a univariate polynomial. Then, the number of queries of the PCPP is only two, but symbols of the oracle proof belong to a large alphabet $\mathbb{F}_q^d$. Similarly, restrictions to affine subspaces of higher dimensions have also been considered, such as the plane-versus-plane test [RS97, MR08] and cube-versus-cube test [BDN17]. The number of field elements needed to be queried is at least linear in d.

Most results apply to polynomials over fields that are larger than the degree bound d. The local testability of Reed-Muller codes when the degree is larger than the field size has been studied in [AKK+03, AKK+05, JPRZ04, KR04]. Aformentioned results show that generalized Reed-Muller codes are locally testable, and query complexity increases as the size of the field decreases.

Note however all the above constructions do not apply to the setting we consider where the function f has domain L^m where L is strictly contained in $\mathbb{F}_q$. Indeed, in such case, the notion of affine subspace does not exist.

By working in the IOPP model, we are able to construct a low-degree test for total degree with strictly linear oracle proof length, which can be generated in linear time and admit logarithmic query complexity and verification time (when the number of variable is considered to be constant). As for the individual degree case, we use a folding approach instead of checking restrictions of the function being tested. As mentioned above, previous works require the verifier to make a number of queries which is at least linear in d. Moreover, the size of the oracle proof [RS92] is polynomial in q^m. In order to further reduce the proof size, constructions using a smaller subset of lines have been investigated [GS02, BSVW03, MR08]. However, such constructions do not achieve a strictly linear oracle proof length, but only proofs of almost linear size. Needlessly to say that proof length is a lower bound on prover running time.

4.2 Preliminaries about multivariate polynomials

4.2.1 Low-degree extensions

To benefit from the algebraic structure of an evaluation code $C \subset \mathbb{F}_q^D$, it is classical to recover a polynomial which coincides with f on D for any $f \in C$. We choose such a polynomial to have low degree with respect to the size of the domain D, when D is a cartesian product.

Proposition 4.1 (Low-degree extension ([BFLS91])). *Let $H_1, \ldots, H_m \subseteq \mathbb{F}_q$ and let $f : H_1 \times \cdots \times H_m \to \mathbb{F}_q$ be a function. Then there exists a unique polynomial $\widehat{f}$ in m variables over $\mathbb{F}_q$ such that :*

1. *$\widehat{f}$ has degree $\deg_{X_i} \widehat{f} < |H_i|$ in its i-th variable,*

2. $\widehat{f}$ *agrees with* f *on* $H_1 \times \cdots \times H_m$.

The polynomial $\widehat{f}$ *is referred to as* the low-degree extension *of the function* f *(with respect to* $\mathbb{F}_q$ *and* $H_1, \ldots, H_m$*).*

Proof. For $H \subset \mathbb{F}_q$ and $h \in H$, denote $L_{H,h}(X) \coloneqq \prod_{k \in H \setminus \{h\}} \frac{X-k}{h-k}$ the Lagrange polynomial. The existence follows from the observation that the polynomial defined by

$$\sum_{\boldsymbol{h} \in H_1 \times \cdots \times H_m} f(\boldsymbol{h}) \prod_{j=1}^{m} L_{H_j,h_j}(X_j)$$

has degree less than $|H_j|$ in each variable and agrees with f on $H_1 \times \cdots \times H_m$. An easy induction on m leads to uniqueness. □

The arithmetic complexity of solving the interpolation problem of computing the coefficients of the low-degree extension of a function $f : H_1 \times \cdots \times H_m \to \mathbb{F}_q$ appears in [Pan94] for general subsets $H_1, \ldots, H_m \subset \mathbb{F}_q$. In our work, we will be specifically interested in the cost of interpolating and evaluating low-degree extensions of a function defined on a grid of size 2^m.

Definition 4.2. *A* multilinear polynomial *is a multivariate polynomial whose degree in each variable is at most one.*

Lemma 4.3 (Multilinear interpolation ([Pan94]))**.** *Let* $H_1, \ldots, H_m \subset \mathbb{F}_q$ *of size 2 and let* $f : H_1 \times \cdots \times H_m \to \mathbb{F}_q$ *be a function. The low-degree extension of* f *is a multilinear polynomial* $\widehat{f} \in \mathbb{F}_q[\boldsymbol{X}]$*. The number of operations required to interpolate* $\widehat{f}$ *is at most* $5m2^{m-1}$ *arithmetic operations.*

Lemma 4.4 (Single-point evaluation of multilinear extension)**.** *Let* $H_1, \ldots, H_m \subset \mathbb{F}_q$ *of size 2 and let* $f : H_1 \times \cdots \times H_m \to \mathbb{F}_q$ *be a function. The low-degree extension of* f *is a multilinear polynomial* $\widehat{f} \in \mathbb{F}_q[\boldsymbol{X}]$ *and, given* $\boldsymbol{p} \in \mathbb{F}_q^m$*, evaluating* $\widehat{f}$ *at* $\boldsymbol{p}$ *can be done in less than* $4(2^m + m)$ *arithmetic operations.*

Proof. For any $\boldsymbol{h} = (h_1, \ldots, h_m) \in H_1 \times \cdots \times H_m$, define $L_{\boldsymbol{h}}(\boldsymbol{X}) \coloneqq \prod_{j=1}^{m} L_{H_j,h_j}(X_j)$. For any $\boldsymbol{p} = (p_1, \ldots, p_m) \in \mathbb{F}_q^m$, we have

$$\widehat{f}(\boldsymbol{p}) = \sum_{\boldsymbol{h} \in H_1 \times \cdots \times H_m} f(\boldsymbol{h}) L_{\boldsymbol{h}}(\boldsymbol{p}). \tag{4.1}$$

As suggested by [VSBW13] regarding multilinear extensions over the boolean hypercube, we observe that $(L_{\boldsymbol{h}}(\boldsymbol{p}))_{\boldsymbol{h} \in H_1 \times \cdots \times H_m}$ can be computed in linear time and linear space using dynamic programming.

Notice that for all $k \in [\![1, m]\!]$,

$$\prod_{j=1}^{k} L_{H_j,h_j}(p_j) = L_{H_k,h_k}(p_k) \prod_{j=1}^{k-1} L_{H_j,h_j}(p_j)$$

and $\deg L_{H_k,h_k} = 1$. Given a table of values containing $\prod_{j=1}^{k-1} L_{H_j,h_j}(p_j)$ for all $(h_1, \ldots, h_{k-1}) \in H_1 \times \cdots \times H_{k-1}$, one can get the values $\prod_{j=1}^{k} L_{H_j,h_j}(p_j)$ for all $(h_1, \ldots, h_k) \in H_1 \times \cdots \times H_k$ by computing the couple of values $\left(L_{H_k,h_k}(p_k)\right)_{h_k \in H_k}$ using 2 additions and 2 divisions, and multiplying both of them by all the 2^{k-1} precomputed values. In sum, this step requires $2^k + 4$ operations. Thus, computing $L_{\boldsymbol{h}}(\boldsymbol{p})$ for all $\boldsymbol{h} \in H_1 \times \cdots \times H_m$ takes $\sum_{j=1}^{m} \left(2^j + 4\right) < 2 \cdot 2^m + 4m$ arithmetic operations. Finally, given the table of values of f and $(L_{\boldsymbol{h}}(\boldsymbol{p}))_{\boldsymbol{h} \in H_1 \times \cdots \times H_m}$, computing the right-hand side of (4.1) takes 2^m multiplications and $(2^m - 1)$ additions. □

4.2.2 Multivariate polynomial decomposition

In order to construct folding operators for multivariate polynomial codes, we give some suitable decomposition of polynomials.

Lemma 4.5. *Let R be an integral domain, and let $q \in R[X]$ be a monic univariate polynomial of degree l. For every $f \in R[\boldsymbol{X}]$ there exists a unique sequence $(f_{\boldsymbol{u}})_{\boldsymbol{u}\in\boldsymbol{U}}$ of polynomials in $R[\boldsymbol{X}]$ such that*

$$f(\boldsymbol{X}) = \sum_{\boldsymbol{u}=(u_1,\dots,u_m)\in\boldsymbol{U}} f_{\boldsymbol{u}}(X_1,\dots,X_m)q(X_1)^{u_1}\cdots q(X_m)^{u_m}, \tag{4.2}$$

where

$$\boldsymbol{U} = \left[\!\left[0, \left\lfloor \deg_{X_1} f/l \right\rfloor\right]\!\right] \times \cdots \times \left[\!\left[0, \left\lfloor \deg_{X_m} f/l \right\rfloor\right]\!\right]$$

and $\deg_{X_i} f_{\boldsymbol{u}}(\boldsymbol{X}) < l$ for each $i \in [\![1,m]\!]$ and $\boldsymbol{u} \in \boldsymbol{U}$.

Proof. The proof is done by induction on the number m of indeterminates, the case $m = 1$ being established in Lemma 2.8. Suppose the result holds for $m-1$ indeterminates and consider $f(\boldsymbol{X})$ as a polynomial in $R[X_1][X_2,\dots,X_m]$. Since $R[X_1]$ is an integral domain, we can apply the induction hypothesis, and there exists a unique sequence

$$(f_{\boldsymbol{u}'}(X_1,X_2,\dots,X_m))_{\boldsymbol{u}'\in\boldsymbol{U}'} \in R[X_1][X_2,\dots,X_m]$$

such that

$$f(X_1,X_2,\dots,X_m) = \sum_{(u_2,\dots,u_m)\in\boldsymbol{U}'} f_{u_2,\dots,u_m}(X_1,X_2,\dots,X_m)q(X_2)^{u_2}\cdots q(X_m)^{u_m}$$

where

$$\boldsymbol{U}' = \left[\!\left[0, \left\lfloor \deg_{X_2} f/l \right\rfloor\right]\!\right] \times \cdots \times \left[\!\left[0, \left\lfloor \deg_{X_m} f/l \right\rfloor\right]\!\right]$$

and, for each $i \in [\![2,m]\!]$:

$$\deg_{X_i} f_{u_2,\dots,u_m}(X_1,X_2,\dots,X_m) < l.$$

Writing

$$f_{u_2,\dots,u_m} = \sum_{0\leqslant u_2,\dots,u_m<l} g_{u_2,\dots,u_m}(X_1)X_2^{u_2}\cdots X_m^{u_m}$$

and applying Lemma 2.8 to each polynomial $g_{u_2,\dots,u_m} \in R[X_1]$, we obtain a unique sequence

$$(g_{u_1,u_2,\dots,u_m}(X_1))_{0\leqslant u_1\leqslant \left\lfloor \deg_{X_1} f/l \right\rfloor}$$

of polynomials in $R[X_1]$ such that

$$g_{u_2,\dots,u_m}(X_1) = \sum_{u_1=0}^{\left\lfloor \deg_{X_1} f/l \right\rfloor} g_{u_1,u_2,\dots,u_m}(X_1)q(X_1)^{u_1}$$

and $\deg g_{u_1,u_2,\dots,u_m}(X_1) < l$. This gives

$$f_{u_2,\dots,u_m} = \sum_{0\leqslant u_2,\dots,u_m<l} \sum_{u_1=0}^{\left\lfloor \deg_{X_1} f/l \right\rfloor} g_{u_1,u_2,\dots,u_m}(X_1)X_2^{u_2}\cdots X_m^{u_m}q(X_1)^{u_1},$$

which leads to the expected decomposition after collecting terms. □

Proposition 4.6 (Multivariate decomposition)**.** *Let R be an integral domain, and let $q \in R[X]$ be a monic univariate polynomial of degree l. For every $f \in R[\boldsymbol{X}]$ there exists a unique sequence $(g_{\boldsymbol{e}})_{\boldsymbol{e} \in [\![0,l-1]\!]^m}$ of polynomials in $R[\boldsymbol{X}]$ such that*

$$f(\boldsymbol{X}) = \sum_{\boldsymbol{e} \in [\![0,l-1]\!]^m} \boldsymbol{X}^{\boldsymbol{e}} g_{\boldsymbol{e}}\left(q(X_1), \ldots, q(X_m)\right), \tag{4.3}$$

and

- *for all $\boldsymbol{e} \in [\![0, l-1]\!]^m$ and $j \in [\![1, m]\!]$, $\deg_{X_j} g_{\boldsymbol{e}} \leqslant \left\lfloor \frac{\deg_{X_j} f}{l} \right\rfloor$,*
- *for all $\boldsymbol{e} \in [\![0, l-1]\!]^m$, $\deg g_{\boldsymbol{e}} \leqslant \left\lfloor \frac{\deg f - \mathsf{w}_{\mathsf{H}}(\boldsymbol{e})}{l} \right\rfloor$.*

Proof. We use the notations of Lemma 4.5. For each $\boldsymbol{u} \in \boldsymbol{U}$, writing each polynomial $f_{\boldsymbol{u}}$ as

$$f_{\boldsymbol{u}}(\boldsymbol{X}) = \sum_{\boldsymbol{e} \in [\![0,l-1]\!]^m} a_{\boldsymbol{u},\boldsymbol{e}} \boldsymbol{X}^{\boldsymbol{e}},$$

Equation (4.2) becomes

$$\begin{aligned} f(\boldsymbol{X}) &= \sum_{\boldsymbol{u} \in \boldsymbol{U}} \sum_{\boldsymbol{e} \in [\![0,l-1]\!]^m} a_{\boldsymbol{u},\boldsymbol{e}} \boldsymbol{X}^{\boldsymbol{e}} q(X_1)^{u_1} \cdots q(X_m)^{u_m}, \\ &= \sum_{\boldsymbol{e} \in [\![0,l-1]\!]^m} \boldsymbol{X}^{\boldsymbol{e}} \sum_{\boldsymbol{u} \in \boldsymbol{U}} a_{\boldsymbol{u},\boldsymbol{e}} q(X_1)^{u_1} \cdots q(X_m)^{u_m}. \end{aligned}$$

For each $\boldsymbol{e} \in [\![0, l-1]\!]^m$, define

$$g_{\boldsymbol{e}}(\boldsymbol{X}) \coloneqq \sum_{\boldsymbol{u} \in \boldsymbol{U}} a_{\boldsymbol{u},\boldsymbol{e}} \boldsymbol{X}^{\boldsymbol{u}}.$$

We thus get the decomposition of Equation (4.3). The bounds for individual degrees of each $g_{\boldsymbol{e}}$ comes from the definition of $\boldsymbol{U}$. Moreover, we have

$$\deg f = \max_{\boldsymbol{e}} \left\{ \deg(\boldsymbol{X}^{\boldsymbol{e}} g_{\boldsymbol{e}}(q(X_1), \ldots, q(X_m))) \right\},$$

thus $\deg f \geqslant \mathsf{w}_{\mathsf{H}}(\boldsymbol{e}) + l \deg g_{\boldsymbol{e}}$.

The uniqueness of the sequence of polynomials $(g_{\boldsymbol{e}})_{\boldsymbol{e}}$ follows from the one of the sequence of polynomials $(f_{\boldsymbol{u}})_{\boldsymbol{u}}$. □

4.3 A first attempt to construct IOPP for tensor products of Reed-Solomon codes

Based on the decomposition given in Proposition 4.6, we present a first construction for tensor product codes, then we will show how efficiency parameters can be improved by increasing the number of rounds and defining the folding operators differently.

4.3.1 Sequence of codes with length divided by 2^m

Let k be a power of two and set $r = \log k$. As for Reed-Solomon codes, we work in the two settings presented in Section 2.1.4. Depending on whether we are in Case 2.1 or Case 2.2, consider $L \subset \mathbb{F}_q$ of size $|L| > k$ which is either a cyclic group of order a power of two, or an additive subgroup of $\mathbb{F}_q$. We will use the notations introduced in Section 2.1.4 and will consider $L_0 = L, L_1, \ldots, L_r$ as defined there.

Set $k_0 \coloneqq k$. For $0 < i \leqslant r$, define $k_{i+1} \coloneqq \frac{k_i}{2}$. In particular, for all i, we have $k_i < |L_i|$. In the sequel, we consider a sequence of tensor products of RS codes $(\mathsf{RS}_i^m)_{0 \leqslant i \leqslant r}$, where RS_i^m is a shorthand notation for $\mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right]^{\otimes m}$, regardless we are in Case 2.1 or Case 2.2.

Notice that, for all $i \in [0, r]$, we have $k_i < |L_i|$. Moreover, each code RS_i^m has same rate $R \coloneqq \left(\frac{k}{|L|}\right)^m$. We denote λ_i the relative distance of the code RS_i^m, i.e. $\lambda_i = \left(1 - \frac{k_i - 1}{|L_i|}\right)^m$. We have that $\min_i \lambda_i \geqslant \left(1 - \frac{k}{|L|}\right)^m$.

4.3.2 Folding operators locally computable from 2^m queries

For each code RS_i^m, $0 \leqslant i < r$, we define a family of folding operators satisfying the distance preservation property. They will enable us to iteratively reduce the problem of proximity testing to a code RS_i^m to a problem of size 2^m times smaller, namely proximity testing to RS_{i+1}^m.

Definition 4.7 (Folding operators). *Let $i \in [0, r-1]$. Let $f \colon L_i^m \to \mathbb{F}_q$ be an arbitrary function and let $\widehat{f}$ be its low-degree extension. Let $(\widehat{g}_e)_{e \in \{0,1\}^m}$ be the 2^m m-variate polynomials provided by Proposition 4.6 applied to $\widehat{f}$. We consider their evaluations on L_{i+1}^m, respectively denoted by g_e. For any $z \in \mathbb{F}_q^m$, we define the* folding of f **Fold** $[f, z]$ *as the following function:*

$$\begin{aligned} \textbf{Fold}\left[f, z\right] : L_{i+1}^m &\to \mathbb{F}_q \\ y &\mapsto \sum_{e \in \{0,1\}^m} z^e g_e(y). \end{aligned}$$

First, we show that this defines a folding operator for the code RS_i^m as per Definition 3.1.

Lemma 4.8 (Completeness). *For any $z \in \mathbb{F}_q^m$, if $f \in \mathsf{RS}_i^m$, then* **Fold** $[f, z] \in \mathsf{RS}_{i+1}^m$.

Proof. Proposition 4.6 shows that, for all $e \in \{0,1\}^m$ and all $j \in [1, m]$, $\deg_{X_j} \widehat{g}_e \leqslant \left\lfloor \frac{k_i - 1}{2} \right\rfloor$, which is strictly less than k_{i+1} since k_i is even. □

Lemma 4.9 (Locality). *Let $f \colon L_i^m \to \mathbb{F}_q$ be an arbitrary function and let $z \in \mathbb{F}_q^m$. The value of* **Fold** $[f, z]$ *at any $y \in L_{i+1}^m$ can be computed with exactly 2^m queries to f.*

Proof. Take $y = (y_1, \ldots, y_m) \in L_{i+1}^m$. For each $j \in [\![1, m]\!]$, define $S_{y_j} \subset L_i$ the coset of A_i (defined in Section 2.1.4) such that $q_i(S_{y_j}) = y_j$ (*i.e.* S_{y_j} is the set of roots of the polynomial $q_i(X) - y_j$). Set

$$S_y \coloneqq \prod_{j=1}^{m} S_{y_j}$$

and consider $P_{f,y} \in \mathbb{F}_q[X]$ the unique low-degree extension of $f_{|S_y}$.

Our goal is to show that for all $z \in \mathbb{F}_q^m$, we have

$$P_{f,y}(z) = \textbf{Fold}\left[f, z\right](y),$$

since it shall imply that the value of $\textbf{Fold}\left[f, z\right](y)$ can be computed by interpolating the set of points $\{(x, f(x)),\ x \in S_y\}$ of size 2^m. By Lemma 4.5, one can write

$$\widehat{f}(X) = \sum_{u \in U} \widehat{f_u}(X) q_i(X_1)^{u_1} \cdots q_i(X_m)^{u_m}$$

with for all $u \in U$ and $j \in [1, m]$, $\deg_{X_j} \widehat{f_u} < 2$. Since the polynomial $\widehat{f}(X)$ and $P_{f,y}(X)$ agree on S_y, we get that

$$\widehat{f}(X) = P_{f,y}(X) \mod (q_i(X_1) - y_1, \ldots, q_i(X_m) - y_m)$$

(this follows for instance from Lemma 1.21). By definition of the low-degree extension, we have $\deg_{X_j} P_{f,y} < 2$ for all $j \in [\![1, m]\!]$, thus

$$P_{f,y}(X) = \sum_{u \in U} \widehat{f_u}(X) y^u.$$

For each $u \in U$, write each polynomial $\widehat{f_u} \in \mathbb{F}_q[X]$ as

$$\widehat{f_u}(X) = \sum_{e \in \{0,1\}^m} a_{u,e} X^e.$$

Proof and result of Proposition 4.6 shows that each polynomial $\widehat{g_e} \in \mathbb{F}_q[X]$ which appears in Definition 4.7 is equal to $\sum_{u \in U} a_{u,e} X^u$. Therefore, for all $y \in L_{i+1}^m$, we have

$$P_{f,y}(X) = \sum_{e \in \{0,1\}^m} X^e \widehat{g_e}(y).$$

Finally, for all $z \in \mathbb{F}_q^m$ and $y \in L_{i+1}^m$, the evaluation of $\textbf{Fold}\left[f, z\right]$ at y can be obtained by evaluating $P_{f,y}$ at z.

□

Let us now show that Definition 4.7 satisfies distance preservation (Definition 3.4).

Proposition 4.10 (Distance preservation)**.** *Let $f_i : L_i^m \to \mathbb{F}_q$ be an arbitrary function. Let $\varepsilon \in \left(0, \frac{1}{3}\right)$ and $\delta < 1 - (1 - \lambda_{i+1} + \varepsilon)^{\frac{1}{3}}$, where λ_{i+1} denotes the relative distance of RS_{i+1}^m. Let $\varphi_i : L_i^m \to [0,1]$ and $\varphi_{i+1} : L_{i+1}^m \to [0,1]$ be weight functions such that*

$$\forall y \in L_{i+1}^m,\ \varphi_{i+1}(y) \leqslant \frac{1}{2^m} \sum_{x \in S_y} \varphi_i(x).$$

If $f : L_i^m \to \mathbb{F}_q$ has weighted agreement $\mathsf{agree}_{\varphi_i}(f, \mathsf{RS}_i^m) < 1 - \delta$, then

$$\Pr_{z \in \mathbb{F}_q^m}\left[\mathsf{agree}_{\varphi_{i+1}}(\textbf{Fold}\left[f, z\right], \mathsf{RS}_{i+1}^m) > 1 - \delta + m\varepsilon\right] < \frac{2m}{\varepsilon^2 q}.$$

Proof. We proceed by contraposition and we assume

$$\Pr_{z\in\mathbb{F}_q^m}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}\left[f,z\right],\mathsf{RS}_{i+1}^m) > 1-\delta+m\varepsilon\right] \geqslant \frac{2m}{\varepsilon^2 q}.$$

Applying Proposition 3.13 on $\mathbf{Fold}\left[f,z\right] = \sum_{e\in\{0,1\}^m} z^e g_e$, we get that there exist $T \subset L_{i+1}^m$ and $(v_e)_{e\in\{0,1\}^m}$, $v_e \in \mathsf{RS}_{i+1}^m$, satisfying

- $\sum_{y\in T} \varphi_{i+1}(y) \geqslant (1-\delta)\left|L_{i+1}^m\right|$,
- for all $e \in \{0,1\}^m$, $g_{e|T} = v_{e|T}$.

For each $e \in \{0,1\}^m$, let us consider $\widehat{v}_e \in \mathbb{F}_q[X]$ the polynomial of individual degrees less than k_{i+1} associated with the codeword $v_e \in \mathsf{RS}_{i+1}^m$. Let R be the polynomial defined by

$$R(X) \coloneqq \sum_{e\in\{0,1\}^m} X^e \widehat{v}_e(q_i(X_1),\ldots,q_i(X_m))$$

and v be the evaluation of R on L_i^m.

Since $k_{i+1} \leqslant k_i/2$, we have $\deg_{X_j} R \leqslant 1 + 2\cdot(k_{i+1}-1) < k_i$, hence $v \in \mathsf{RS}_i^m$. For all $y \in T$ and $x \in S_y$, i.e. $\pi(x) = y$, $v(x) = \sum_{e\in\{0,1\}^m} x^e v_e(\pi(x))$ and

$$f(x) = \sum_{e\in\{0,1\}^m} x^e \widehat{g}_e(y) = \sum_{e\in\{0,1\}^m} x^e g_e(y) = \sum_{e\in\{0,1\}^m} x^e v_e(y) = v(x). \tag{4.4}$$

Thus v agrees with f on $S_T \coloneqq \bigsqcup_{y\in T} S_y$. Since $v \in \mathsf{RS}_i^m$, we have

$$\mathsf{agree}_{\varphi_i}(f,\mathsf{RS}_i^m) \geqslant \frac{1}{\left|L_i^m\right|}\sum_{x\in S_T}\varphi_i(x) = \frac{1}{\left|L_i^m\right|}\sum_{y\in T}\sum_{x\in S_y}\varphi_i(x) \geqslant \frac{1}{\left|L_{i+1}^m\right|}\sum_{y\in T}\varphi_{i+1}(y).$$

Eventually, we conclude that $\mathsf{agree}_{\varphi_i}(f,\mathsf{RS}_i^m) \geqslant 1-\delta$ by definition of T. This contradicts the hypothesis on f. □

4.3.3 IOPP for tensor product of RS codes

Given a sequence of codes $(\mathsf{RS}_i^m)_{0\leqslant i\leqslant r}$ as defined in Section 4.3.1 and a family of folding operators for each code RS_i^m (see Section 4.3.2), the generic construction described in Section 3.1.2 leads to a public-coin IOPP $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ for the code RS_0^m.

Notice that the last function f_r is supposed to be constant. Therefore, we use the variant of the protocol described in Remark 3.7. Specifically, instead of sending f_r during the COMMIT phase, the prover $\mathscr{P}_{\mathsf{RS}^m}$ sends a single field element $\beta \in \mathbb{F}_q$. The verifier $\mathscr{V}_{\mathsf{RS}^m}$ does not run a membership test to C_r but checks the equation $\beta = \mathbf{Fold}\left[f_{r-1}, z_{r-1}\right](y_r)$.

The properties of the resulting IOPP system $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ are displayed in the following theorem.

Theorem 4.11. *Let k, m be positive integers such that $k > 1$ is a power of two. Let $L \subset \mathbb{F}_q^\times$ as described in Section 2.1.4 such that $k < |L|$. Then, the generic construction of Section 3.1.2 leads to public-coin IOPP system $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ for the tensor product code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$ of blocklength n^m satisfying:*

Completeness: *If $f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$ and if the oracles $f_1, \dots f_r$ are computed by an honest prover $\mathscr{P}_{\mathsf{RS}^m}$, then $\mathscr{V}_{\mathsf{RS}^m}$ outputs* ***accept*** *with probability 1.*

Soundness: *Assume that $f : L^m \to \mathbb{F}_q$ is δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$. Define $\lambda \coloneqq \left(1 - \frac{k}{|L|}\right)^m$ and, for any $\varepsilon \in \left(0, \frac{1}{3}\right)$, set $\gamma(\lambda, \varepsilon) \coloneqq 1 - (1 - \lambda + \varepsilon)^{1/3}$. Then, for any unbounded prover $\mathscr{P}^*$, the verifier $\mathscr{V}_{\mathsf{RS}^m}$ outputs* accept *after α repetitions of the* QUERY *phase with probability at most*

$$\frac{2m \log k}{\varepsilon^2 q} + (1 - \min(\delta, \gamma(\varepsilon, \lambda)) + \varepsilon m \log k)^\alpha.$$

Moreover, $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ has the following properties:

- *round complexity* $\mathsf{r} = \log k$,
- *proof length* $\mathsf{l} < \frac{n^m}{2^m - 1}$,
- *query complexity* $\mathsf{q} = \alpha 2^m \log k + 1$,
- *prover complexity* $\mathsf{tp} < 4(m+2)n^m$,
- *verifier complexity* $\mathsf{tv} < 4\alpha(2^m + m)\log k$.

Proof. We apply the construction of the public-coin IOPP system presented in Section 3.1.2 with the family of folding operators defined in Section 4.3.2. Completeness and soundness follow from Theorem 3.8, recalling that the minimum distances of the codes $\mathsf{RS}_0^m, \dots, \mathsf{RS}_{r-1}^m$ are all greater than $\left(1 - \frac{k}{|L|}\right)^m$. The number of round is $r = \log k < \log |L|$ by definition. For a single repetition of the query test, $\mathscr{V}_{\mathsf{RS}^m}$ queries each oracle f_i, $i \in [\![0, r-1]\!]$, at 2^m locations. The verifier retrieves β a single time, which yields the claimed query complexity.

The total proof length is

$$\sum_{i=1}^{r} |L_i^m| = \sum_{i=1}^{r} \frac{n^m}{2^{mi}} < \frac{n^m}{2^m - 1}.$$

We examine prover complexity. Let $f : L_i^m \to \mathbb{F}_q$ and $\boldsymbol{z} \in \mathbb{F}_q^m$. For each $\boldsymbol{y} \in L_{i+1}^m$, the prover evaluates the low-degree extension $P_{f,\boldsymbol{y}}(\boldsymbol{X})$ of $f_{|S_{\boldsymbol{y}}}$ at $\boldsymbol{z}$, where $S_{\boldsymbol{y}} = \pi_i^{-1}(\{\boldsymbol{y}\})$. It follows from Lemma 4.4 that the number of operations to evaluate $\mathbf{Fold}\left[f, \boldsymbol{z}\right]$ on L_{i+1}^m is $4(2^m + m)\left|L_{i+1}^m\right|$. We deduce that the cost of honestly generating $\mathscr{P}_{\mathsf{RS}^m}$'s messages is

$$\sum_{i=1}^{r} 4(2^m + m)\left|L_{i+1}^m\right| < 4(2^m + m)\frac{n^m}{2^m - 1} \leqslant 4(m+2)n^m.$$

We also deduce from Lemma 4.4 that the verifier complexity is less than $\alpha \sum_{i=1}^{r} 4(2^m + m)$. □

4.4 IOPP for tensor product of RS codes by folding with respect to each variable

The construction presented in this section essentially consists in applying the FRI protocol to each variable. We use the possibilty of folding with respect to a single indeterminate, instead of folding along all the indeterminates at once. We call this *partial folding*.

Let us first present the idea for the case where $m = 2$ and L is a multiplicative subgroup of a field of odd characteristic. Given a function $f : L^2 \to \mathbb{F}_q$, and $\hat{f}(X_1, X_2)$ its associated low-degree

extension, we can decompose it as

$$\hat{f}_1(X_1, X_2) = \hat{g}_0(X_1^2, X_2) + X_1\hat{g}_1(X_1^2, X_2).$$

For $a \in \mathbb{F}$, the notation $\mathbf{Fold}_{X_1}[f, a] : q(L) \times L \to \mathbb{F}_q$ will refer to the function whose low-degree extension is the polynomial

$$\hat{g}_0(X_1, X_2) + z\hat{g}_1(X_1, X_2).$$

As for the univariate case (Section 2.1.5), for any $y \in q(L)$ and $a \in \mathbb{F}_q$, one can compute $\mathbf{Fold}_{X_1}[f, a](y)$ from exactly two entries of f. Such a folding operator $\mathbf{Fold}_{X_1}[\cdot, \cdot]$ will allow us to reduce a problem of proximity to a code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes 2}$ to a similar but smaller problem, which is associated to the code

$$\mathsf{RS}\left[\mathbb{F}_q, q(L), k/2\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L, k\right].$$

Remark 4.12. *We can also write*

$$\hat{f}(X_1, X_2) = \hat{h}_0(X_1, X_2^2) + X_2\hat{h}_1(X_1, X_2^2),$$

and given $b \in \mathbb{F}_q$, we can define $\mathbf{Fold}_{X_2}[f, b] : L_1 \times q(L_2) \to \mathbb{F}_q$ whose low-degree extension is $\hat{h}_0(X_1, X_2) + b\hat{h}_1(X_1, X_2)$. A simple calculation shows that partial folding admits a "commutative property", namely for $f : L_1 \times L_2 \to \mathbb{F}_q$, and $\boldsymbol{z} = (a, b) \in \mathbb{F}^2$, we have

$$\mathbf{Fold}[f, \boldsymbol{z}] = \mathbf{Fold}_{X_2}[\mathbf{Fold}_{X_1}[f, a], b] = \mathbf{Fold}_{X_1}[\mathbf{Fold}_{X_2}[f, b], a]. \tag{4.5}$$

Let us now assume that we want to construct an IOPP for a m-wise tensor product of Reed-Solomon code $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes m}$. The idea of the construction is to start by folding with respect to the first variable, until a sufficiently small degree with respect to X_1 is reached. We use $s = \log k_0$ rounds of interaction to reduce the problem of proximity for $\mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes m}$ to a problem of proximity for

$$\mathsf{RS}\left[\mathbb{F}_q, L_s, k_s\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes(m-1)}.$$

Then, we repeat the process with respect to the second variable, using $s = \log k_0$ rounds of interactions to reduce the proximity problem for $\mathsf{RS}\left[\mathbb{F}_q, L_s, k_s\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes(m-1)}$ to a proximity problem for $\mathsf{RS}\left[\mathbb{F}_q, L_s, k_s\right]^{\otimes 2} \otimes \mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes(m-2)}$.

By repeating this process for the remaining indeterminates, namely after a total number of rounds $r \coloneqq m \log k_0$, we are left with a trivial proximity problem for the code

$$\mathsf{RS}\left[\mathbb{F}_q, L_s, k_s\right]^{\otimes m}.$$

Compared to Section 4.3, this approach yields an IOPP for tensor product of Reed-Solomon codes that has the exact same efficiency parameters than the FRI protocol, which deals with the univariate case. In particular, even when considering that the number of variables m is not constant, we achieve strictly linear prover time and strictly logarithmic verification time.

In order not to overload notations, we present our IOPP construction for a m-wise tensor product of Reed-Solomon codes. One can readily verify that the construction we are going to present can be generalized to the case where the initial proximity testing problem is concerned with a tensor product of distinct Reed-Solomon codes, namely

$$\mathsf{RS}\left[\mathbb{F}_q, L^{(1)}, k^{(1)}\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L^{(2)}, k^{(2)}\right] \otimes \cdots \otimes \mathsf{RS}\left[\mathbb{F}_q, L^{(m)}, k^{(m)}\right]$$

with different degree bounds $\left(k^{(j)}\right)_{1 \leqslant j \leqslant m}$ and supports $\left(L^{(j)}\right)_{0 \leqslant j \leqslant m}$.

4.4.1 Sequence of codes with length divided by 2

Once again, we assume that $L_0, \dots, L_r \subset \mathbb{F}_q$ satisfy Case 2.1 or Case 2.2 of Section 2.1.4. Let k be a power of two and set $s \coloneqq \log k$. As in Section 4.3.1, we set $L_0 \coloneqq L$ and $k_0 \coloneqq k$. For any $i \in [\![0, s-1]\!]$, $L_{i+1} = q_i(L_i)$, where q_i is defined in Section 2.1.4, and $k_{i+1} \coloneqq \frac{k_i}{2}$. Letting $r \coloneqq ms$, we define a sequence of $r+1$ codes $C_0, C_1, \dots, C_r$ as follows. The first code C_0 is $C_0 \coloneqq \mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes m}$. For $j \in [\![1, m]\!]$ and $i \in [\![0, s-1]\!]$, the code $C_{(j-1)s+i}$ is

$$C_{(j-1)s+i} \coloneqq \mathsf{RS}\left[\mathbb{F}_q, L_s, k_s\right]^{\otimes j-1} \otimes \mathsf{RS}\left[\mathbb{F}_q, L_i, k_i\right] \otimes \mathsf{RS}\left[\mathbb{F}_q, L_0, k_0\right]^{\otimes m-j},$$

where we use the convention that, for any $\mathbb{F}_q$-linear space V, $V^{\otimes 0} = \mathbb{F}_q$ and $V^{\otimes 1} = V$ (we have $\mathbb{F}_q \otimes V = V \otimes \mathbb{F}_q = V$).

4.4.2 Partial folding operators

Let us fix $j \in [\![1, m]\!]$ and $i \in [\![0, s-1]\!]$ for this subsection. We want to define a folding operator with respect to the j-th variable. Once again, we assume $\deg q_i = 2$ to simplify the exposition. One can readily verify that the arguments presented here can be carried over to the case $\deg q_i \geqslant 2$.

For $z \in \mathbb{F}_q$, we construct a folding operator

$$\mathbf{Fold}_j\left[\cdot, z\right] : \mathbb{F}^{L_s^{j-1} \times L_i \times L_0^{m-j}} \times \mathbb{F} \to \mathbb{F}^{L_s^{j-1} \times L_{i+1} \times L_0^{m-j}}$$

that will allow us to reduce the problem of proximity to the code $C_{(j-1)s+i}$ to a problem of half the size.

For $\boldsymbol{X} = (X_1, X_2, \dots, X_m)$, we denote by $\boldsymbol{X}_{\bar{j}}$ the tuple obtained by removing the j-th entry of $\boldsymbol{X}$, i.e. $\boldsymbol{X}_{\bar{j}} = (X_1, \dots, X_{j-1}, X_{j+1}, \dots, X_m)$. Accordingly, we use the notation $\mathbb{F}_q[\boldsymbol{X}_{\bar{j}}]$ to refer to the polynomial ring $\mathbb{F}_q[X_1, \dots, X_{j-1}, X_{j+1}, \dots, X_m]$. The following corollary is a straightforward generalization of Lemma 2.8.

Corollary 4.13. *Given a polynomial $\widehat{f} \in \mathbb{F}_q[\boldsymbol{X}_{\bar{j}}][X_j]$ and a monic polynomial $q \in \mathbb{F}_q[X]$ of degree 2, there are two polynomials $\widehat{g_0}, \widehat{g_1} \in \mathbb{F}_q[\boldsymbol{X}_{\bar{j}}][X_j]$ such that*

$$\widehat{f}(\boldsymbol{X}_{\bar{j}})(X_j) = \widehat{g_0}(\boldsymbol{X}_{\bar{j}})(q_i(X_j)) + X_j \widehat{g_1}(\boldsymbol{X}_{\bar{j}})(q_i(X_j)), \tag{4.6}$$

and, for all $j' \in [\![1, m]\!]$,

$$\deg_{X_{j'}} \widehat{g_0}, \deg_{X_{j'}} \widehat{g_1} \leqslant \begin{cases} \frac{\deg_{X_{j'}} \widehat{f}}{2} & \text{if } j' = j \\ \deg_{X_{j'}} \widehat{f} & \text{otherwise.} \end{cases}$$

Definition 4.14. *Given $q_i(X)$ defined in Section 2.1.4, we define $\pi_{j,i} \colon \mathbb{F}_q^m \to \mathbb{F}_q^m$ as the function*

$$\pi_{j,i} \colon (x_1, \dots, x_m) \mapsto (x_1, \dots, x_{j-1}, q_i(x_j), x_{j+1}, \dots, x_m).$$

Definition 4.15 (Folding with respect to the j-th variable)**.** *Let $f : L_s^{j-1} \times L_i \times L_0^{m-j} \to \mathbb{F}_q$ be an arbitrary function and $\widehat{f} \in \mathbb{F}_q[\boldsymbol{X}]$ its low-degree extension. Let g_0, g_1 be the evaluations on $L_s^{j-1} \times L_{i+1} \times L_0^{m-j}$ of the polynomials $\widehat{g_0}, \widehat{g_1} \in \mathbb{F}_q[\boldsymbol{X}]$ given by Corollary 4.13, respectively. For $z \in \mathbb{F}_q$, we define the j-th partial folding of f as the function*

$$\mathbf{Fold}_j\left[f, z\right] : L_s^{j-1} \times L_{i+1} \times L_0^{m-j} \to \mathbb{F}_q$$

defined by $\mathbf{Fold}_j\left[f, z\right] \coloneqq g_0 + z g_1$.

An immediate consequence of Definition 4.15 is the following lemma.

Lemma 4.16 (Completeness). *For any $z \in \mathbb{F}_q$, if $f \in C_{(j-1)s+i}$, then $\mathbf{Fold}_j[f,z] \in C_{(j-1)s+i+1}$.*

Lemma 4.17 (Local computability). *Let $f : L_s^{j-1} \times L_i \times L_0^{m-j} \to \mathbb{F}_q$ be an arbitrary function and let $z \in \mathbb{F}_q$. For any $\boldsymbol{y} \in L_s^{j-1} \times L_{i+1} \times L_0^{m-j}$, the value $\mathbf{Fold}_j[f,z](\boldsymbol{y})$ can be computed with exactly 2 entries of f.*

Proof. Let $\boldsymbol{y} \in L_s^{j-1} \times L_{i+1} \times L_0^{m-j}$. Denote $S_{\boldsymbol{y}} \subset L_s^{j-1} \times L_i \times L_0^{m-j}$ the set

$$S_{\boldsymbol{y}} \coloneqq \pi_{j,i}^{-1}(\{\boldsymbol{y}\}).$$

Since $q_i(X)$ has two distinct roots, the set $S_{\boldsymbol{y}}$ has 2 elements. Let us consider $P_{f,\boldsymbol{y}} \in \mathbb{F}_q[X]$ the polynomial of degree less than 2 such that, for all $\boldsymbol{x} \in S_{\boldsymbol{y}}$,

$$P_{f,\boldsymbol{y}}(x_j) = f(\boldsymbol{x}).$$

We have that the polynomial equation

$$P_{f,\boldsymbol{y}}(X) = \widehat{g}_0(\boldsymbol{y}_{\bar{j}})(y_j) + X\widehat{g}_0(\boldsymbol{y}_{\bar{j}})(y_j)$$

holds, since both polynomials have degree less than 2 and agree on two distinct values. Recalling Definition 4.15 we have, for all $z \in \mathbb{F}_q$,

$$P_{f,\boldsymbol{y}}(z) = \mathbf{Fold}_j[f,z](\boldsymbol{y}).$$

In particular, the value $\mathbf{Fold}_j[f,z](\boldsymbol{y})$ can be computed by interpolating the set of points

$$\{(x_j, f(\boldsymbol{x})) \mid \boldsymbol{x} \in S_{\boldsymbol{y}}\}$$

of size two. □

Proposition 4.18 (Distance preservation). *Let $f : L_s^{j-1} \times L_i \times L_0^{m-j} \to \mathbb{F}_q$ be an arbitrary function. Let $\varepsilon \in \left(0, \frac{1}{3}\right)$ and $\delta < 1 - (1 - \Delta(C_{(j-1)s+i+1}) + \varepsilon)^{\frac{1}{3}}$. Let $\varphi_i : L_s^{j-1} \times L_i \times L_0^{m-j} \to [0,1]$ and $\varphi_{i+1} : L_s^{j-1} \times L_{i+1} \times L_0^{m-j} \to [0,1]$ be weight functions such that*

$$\forall \boldsymbol{y} \in L_s^{j-1} \times L_{i+1} \times L_0^{m-j},\ \varphi_{i+1}(\boldsymbol{y}) \leqslant \frac{1}{2} \sum_{\boldsymbol{x} \in S_{\boldsymbol{y}}} \varphi_i(\boldsymbol{x}).$$

If $f : L_s^{j-1} \times L_i \times L_0^{m-j} \to \mathbb{F}_q$ has weighted agreement $\mathsf{agree}_{\varphi_i}(f, C_{(j-1)s+i}) < 1 - \delta$, then

$$\Pr_{z \in \mathbb{F}_q}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}_j[f,z], C_{(j-1)s+i+1}) > 1 - \delta + \varepsilon\right] < \frac{2}{\varepsilon^2 q}.$$

Proof. We proceed by contraposition and we assume

$$\Pr_{z \in \mathbb{F}_q}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}_j[f,z], C_{(j-1)s+i+1}) > 1 - \delta + \varepsilon\right] \geqslant \frac{2}{\varepsilon^2 q}.$$

Applying Lemma 3.14 on $\mathbf{Fold}_j[f,z] = g_0 + zg_1$, we get that there exist $T \subset L_s^{j-1} \times L_{i+1} \times L_0^{m-j}$ and $v_0, v_1 \in C_{(j-1)s+i+1}$, satisfying

- $\sum_{\boldsymbol{y}\in T} \varphi_{i+1}(\boldsymbol{y}) \geqslant (1-\delta)\left|L_s^{j-1} \times L_{i+1} \times L_0^{m-j}\right|$,
- $g_{0|T} = v_{0|T}$ and $g_{1|T} = v_{1|T}$.

Let us consider $\widehat{v_0}, \widehat{v_1} \in \mathbb{F}_q[\boldsymbol{X}]$ the polynomial associated to the codewords $v_0, v_1 \in C_{(j-1)s+i+1}$ respectively. We have:

$$\deg_{X_{j'}} \widehat{v_0}, \deg_{X_{j'}} \widehat{v_1} \begin{cases} < k_s, & \text{for } 1 \leqslant j' < j, \\ < k_i, & \text{for } j' = j, \\ < k_0, & \text{for } j < j' \leqslant m. \end{cases}$$

Let $R \in \mathbb{F}_q[\boldsymbol{X}_{\bar{j}}][X_j]$ be the polynomial

$$R(\boldsymbol{X}_{\bar{j}})(X_j) \coloneqq \widehat{v_0}(\boldsymbol{X}_{\bar{j}})(q_i(X_j)) + X_j v_1(\boldsymbol{X}_{\bar{j}})(q_i(X_j)).$$

Consider v the evaluation on $L_s^{j-1} \times L_i \times L_0^{m-j}$ of $R(\boldsymbol{X})$ viewed as a polynomial in $\mathbb{F}_q[\boldsymbol{X}]$. Since $k_{i+1} \leqslant k_i/2$, we have $\deg_{X_j} R \leqslant 1 + 2 \cdot (k_{i+1} - 1) < k_i$, and thus $v \in C_{(j-1)s+i}$.

For all $\boldsymbol{y} \in T$ and $\boldsymbol{x} \in \pi_{j,i}^{-1}(\{\boldsymbol{y}\})$, we have

$$\begin{aligned} f(\boldsymbol{x}) &= \widehat{g_0}(\pi_{j,i}(\boldsymbol{x})) + x_j \widehat{g_1}(\pi_{j,i}(\boldsymbol{x})) \\ &= g_0(\pi_{j,i}(\boldsymbol{x})) + x_j g_1(\pi_{j,i}(\boldsymbol{x})) \\ &= v_0(\pi_{j,i}(\boldsymbol{x})) + x_j v_1(\pi_{j,i}(\boldsymbol{x})) \\ &= v(\boldsymbol{x}). \end{aligned}$$

Thus v agrees with f on $S_T \coloneqq \bigsqcup_{\boldsymbol{y}\in T} S_{\boldsymbol{y}}$. Since $v \in C_{(j-1)s+i}$, we have

$$\begin{aligned} \mathsf{agree}_{\varphi_i}(f, C_{(j-1)s+i}) &\geqslant \frac{1}{\left|L_s^{j-1} \times L_i \times L_0^{m-j}\right|} \sum_{\boldsymbol{x}\in S_T} \varphi_i(\boldsymbol{x}) \\ &= \frac{1}{\left|L_s^{j-1} \times L_i \times L_0^{m-j}\right|} \sum_{\boldsymbol{y}\in T} \sum_{\boldsymbol{x}\in S_{\boldsymbol{y}}} \varphi_i(\boldsymbol{x}) \\ &\geqslant \frac{1}{\left|L_s^{j-1} \times L_{i+1} \times L_0^{m-j}\right|} \sum_{\boldsymbol{y}\in T} \varphi_{i+1}(\boldsymbol{y}). \end{aligned}$$

Eventually, we conclude that $\mathsf{agree}_{\varphi_i}(f, C_{(j-1)s+i}) \geqslant 1 - \delta$ by definition of T. This contradicts the hypothesis on f. □

4.4.3 Improved IOPP for tensor product of RS codes

Given a sequence of codes $(C_i)_{0\leqslant i\leqslant r}$ as defined in Section 4.4.1 and a family of folding operators for each code C_i as in Section 4.4.2, Construction 3.5 leads to a public-coin IOPP $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ for the code C_0.

With our choices of parameters, the last function f_r is again supposed to be constant. Once again, we use the variant of the protocol described in Remark 3.7. For the last round of the COMMIT phase, the prover $\mathscr{P}_{\mathsf{RS}^m}$ sends a single field element $\beta \in \mathbb{F}_q$ and the verifier checks that

$$\beta = \mathbf{Fold}_j\,[f_{r-1}, z_{r-1}](\boldsymbol{y}_r).$$

This leads to the following theorem.

Theorem 4.19. *Let k, m be positive integers such that $k > 1$ is a power of two. Let $L \subset \mathbb{F}_q^\times$ be a set of size $|L| > k$ defined as per Section 2.1.4. Construction 3.5 with the folding operators defined as per Defintion 4.15 leads to public-coin IOPP system $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ for the tensor product code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$ of blocklength n^m satisfying:*

Completeness: *If $f \in \mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$ and if the oracles $f_1, \dots f_r$ are computed by an honest prover $\mathscr{P}_{\mathsf{RS}^m}$, then $\mathscr{V}_{\mathsf{RS}^m}$ outputs **accept** with probability 1.*

Soundness: *Assume that $f : L^m \to \mathbb{F}_q$ is δ-far from $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]^{\otimes m}$. Define $\lambda \coloneqq \left(1 - \frac{k}{|L|}\right)^m$ and, for any $\varepsilon \in \left(0, \frac{1}{3}\right)$, set $\gamma(\lambda, \varepsilon) \coloneqq 1 - (1 - \lambda + \varepsilon)^{1/3}$. Then, for any unbounded prover $\mathscr{P}^*$, the verifier $\mathscr{V}_{\mathsf{RS}^m}$ outputs accept after α repetitions of the QUERY phase with probability at most*

$$\frac{2m \log k}{\varepsilon^2 q} + (1 - \min(\delta, \gamma(\varepsilon, \lambda)) + \varepsilon m \log k)^\alpha.$$

Moreover, $(\mathscr{P}_{\mathsf{RS}^m}, \mathscr{V}_{\mathsf{RS}^m})$ has the following properties:

- *rounds complexity* $\mathsf{r} = \log k^m$,
- *proof length* $\mathsf{l} < n^m$,
- *query complexity* $\mathsf{q} = 2\alpha \log k^m + 1$,
- *prover complexity* $\mathsf{tp} < 8n^m$,
- *verifier complexity* $\mathsf{tv} < 8\alpha \log k^m$.

Proof. Completeness and soundness follows from Theorem 3.8 by observing that the relative distances of the codes $C_0, \dots, C_r$ are greater than $\left(1 - \frac{k}{|L|}\right)^m$.

By construction, the IOPP has $r = \log(k^m)$ rounds. During the QUERY phase, the verifier makes 2 queries to each function $f_0, \dots, f_{r-1}$. The verifier also queries the element β sent during the last round of the COMMIT phase. This gives the claimed query complexity.

Let us denote by n_i the length of the code C_i for $i \in [\![0, r]\!]$. The proof length is sum of the n_i's for $i \in [\![1, r]\!]$. Since $n_{i+1} = n_i/2$, the sum of the first terms of a geometric sequence gives the claimed proof length.

For fixed $j \in [\![1, m]\!], i \in [\![0, s-1]\!]$ and $\boldsymbol{y} \in L_s^{j-1} \times q_i(L_i) \times L_0^{m-j}$, we compute the number c of field operations to perform in order to get the value $\mathbf{Fold}_j\left[f, z\right](\boldsymbol{y})$. Recall that $\mathbf{Fold}_j\left[f, z\right](\boldsymbol{y})$ can be computed by interpolating the set of points $\left\{(x_j, f(\boldsymbol{y}_{\bar{j}})(x_j)) \mid q_i(x_j) = y_j\right\}$ and evaluating the obtained polynomial at z.

We consider x_j, x_j' the two distinct roots of the polynomial $q_i(X) - y_j$. For

$$\boldsymbol{x} \coloneqq (y_1, \dots, y_{j-1}, x_j, y_{j+1}, \dots, y_m) \text{ and } \boldsymbol{x}' \coloneqq (y_1, \dots, y_{j-1}, x_j', y_{j+1}, \dots, y_m),$$

we have

$$\mathbf{Fold}_j\left[f, z\right](\boldsymbol{y}) = P_{f,\boldsymbol{y}}(z) = \frac{1}{x_j - x_j'}\left(f(\boldsymbol{x})(z - x_j) - f(\boldsymbol{x}')(z - x_j')\right).$$

In this case, computing $\mathbf{Fold}_j\left[f, z\right](\boldsymbol{y})$ takes at most 8 field operations.

At each round, the prover performs $8n_i$ computations. Summing over r rounds, we get the claimed prover complexity. During one round of the QUERY phase, the verifier evaluates $\mathbf{Fold}_j\left[f, z\right]$ at a single point, therefore the verifier complexity for a repetition parameter s is $8\alpha r \leqslant 8\alpha m \log k$. □

Remark 4.20. *In the case where the polynomial q_i has degree larger than 2, one can compute the cost of evaluating the folding of a function at a single point by looking at the number of operations needed to interpolate and evaluate at a single point a polynomial of degree less than $\deg q_i$ (e.g. using Lagrange interpolation formula).*

Comparisons with the univariate case. Soundness of the FRI protocol [BBHR18a] has been analyzed in [BBHR18a, BKS18, BGKS20, BCI+20]. For a Reed-Solomon code of blocklength N, relative distance λ and alphabet $\mathbb{F}_q$ of size *linear* in N, the soundness is given by [BGKS20]. Specifically, the FRI protocol is a r-round IOPP, $\mathsf{r} < \log N$, with soundness error (for a single repetition of the QUERY phase) bounded from above by

$$\frac{2\mathsf{r}}{\varepsilon^2 \left|\mathbb{F}_q\right|} + (1 - \min(\delta, \gamma(\varepsilon, \lambda)) + \varepsilon\mathsf{r}),$$

where $\gamma(\varepsilon, \lambda) = 1 - (1 - \lambda + \varepsilon)^{1/3}$. Authors of [BGKS20] also showed that the bound $\gamma(\varepsilon, \lambda)$ is tight for RS codes evaluated over the entire field, and when this field has characteristic two. Subsequently, [BCI+20] improved soundness of the FRI protocol for quadratic-size fields using symbolic list-decoding algorithms for RS codes.

We point out that the soundness error of our IOPP for tensor product of RS code is given by the exact same formula than the one shown in [BGKS20] for the univariate case, albeit tensor codes have worse relative distance.

In Figure 4.2, we present the parameters of the FRI protocol for RS codes and our IOPP for tensor product of RS codes side by side, for one repetition of the QUERY phase. The parameters for the FRI protocol are taken from Theorem 2.17. We consider codes of blocklength N and dimension K and a single repetition of the QUERY phase. In order to achieve arbitrary constant soundness error, both protocols require to repeat the QUERY phase. This process increases query complexity and verifier running time by a multiplicative factor independent of N. However, the FRI protocol has better soundness, thus requires less repetitions.

Scheme	Prover	Verifier	Query	Length	Rounds
RS IOPP [BBHR18a]	$< 8N$	$< 8 \log K$	$2 \log K + 1$	$< N$	$\log K$
$\mathrm{RS}^{\otimes m}$ IOPP (Thm. 4.11)	$< (2m+4)N$	$< 4\left(\frac{2^m}{m} + 1\right) \log K$	$\frac{2^m}{m} \log K$	$< \frac{N}{2^m - 1}$	$\frac{\log K}{m}$
$\mathrm{RS}^{\otimes m}$ IOPP (Thm. 4.19)	$< 8N$	$< 8 \log K$	$2 \log K + 1$	$< N$	$\log K$

Figure 4.2: Comparison between the IOPP for a RS code of [BBHR18a] and our IOPPs for a tensor product of RS code. We compare codes with the same blocklength N and same dimension K.

4.5 Short Reed-Muller codes

Our IOPP for Short Reed-Muller codes resembles the construction presented in Section 4.3, since the approach of folding according to each variable (Section 4.4) cannot be reproduced.

4.5.1 Sequence of codes

Similarly to Section 4.3.1, we will consider two families of short Reed-Muller codes, depending on whether Case 2.1 or Case 2.2 holds (Section 2.1.4). Let k be a power of two, $k < |L|$ and set $r = \log_2 k$. We consider $L_0 = L, L_1, \dots, L_r$ as constructed in Section 2.1.4.

Set $k_0 \coloneqq k$. For $0 < i \leqslant r$, define $k_{i+1} \coloneqq \frac{k_i}{2}$. In particular, for all i, we have $k_i < |L_i|$. Let us denote by SRM_i the short Reed-Muller code $\mathsf{SRM}\left[\mathbb{F}_q, L_i, m, k_i\right]$.

Starting from the code $\mathsf{SRM}_0 = \mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$, this defines a sequence of Reed-Muller codes $(\mathsf{SRM}_i)_{0\leqslant i\leqslant r}$. For each i, the relative distance λ_i of SRM_i is at least $1 - \frac{k_i - 1}{|L_i|}$, hence $\min_i \lambda_i \geqslant 1 - \frac{k}{|L|}$.

4.5.2 Folding operators

Let $(\mathsf{SRM}_i)_{0\leqslant i\leqslant r}$ be a sequence of short Reed-Muller codes defined as described in Section 4.5.1 (regardless we are in Case 2.1 or Case 2.2). For each $i \in [\![0, r-1]\!]$, we define a family of folding operators which will enables us to iteratively reduce the problem of proximity testing to a code SRM_i to a problem of size 2^m times smaller, namely proximity testing to SRM_{i+1}.

Note that the sequences of evaluation domains $(L_i^m)_i$ and degree bounds $(k_i)_i$ are defined exactly the same way as in the tensor product case. However, if we design folding operators for Reed-Muller codes by following the same construction than in Definition 4.7, then the distance preservation property does not hold anymore. For this reason, some *balancing functions*[1] are involved in the definition of folding operators for Reed-Muller codes (and will also appear in the context of algebraic-geometry codes in Chapter 5)

Definition 4.21 (Balancing functions)**.** *Let $i \in [\![0, r-1]\!]$. For any $\boldsymbol{e} \in \{0,1\}^m$, we call a* balancing function *any map $h_{\boldsymbol{e}} : L_{i+1}^m \to \mathbb{F}_q$ which corresponds to the evaluation of a m-variate multilinear monic monomial $\hat{h}_{\boldsymbol{e}}$ of total degree exactly $\left\lfloor \frac{\mathsf{w_H}(\boldsymbol{e})}{2} \right\rfloor$. We call $(h_{\boldsymbol{e}})_{\boldsymbol{e}\in\{0,1\}^m}$ a* balancing tuple for the code SRM_{i+1}.

Definition 4.22 (Folding operator)**.** *Let $i \in [0, r-1]$. Let $(h_{\boldsymbol{e}})_{\boldsymbol{e}\in\{0,1\}^m}$ be a balancing tuple for SRM_{i+1} and let $f : L_i^m \to \mathbb{F}_q$ be an arbitrary function. Given $(\hat{g}_{\boldsymbol{e}})_{\boldsymbol{e}\in\{0,1\}^m}$ the 2^m m-variate polynomials of the decomposition of Proposition 4.6, denote $g_{\boldsymbol{e}}$ the evaluation on L_{i+1}^m of $\hat{g}_{\boldsymbol{e}}$. For any $(\boldsymbol{z}, \boldsymbol{z}') \in \left(\mathbb{F}_q^m\right)^2$, we define the* folding of f *as the function* $\mathbf{Fold}\left[f, (\boldsymbol{z}, \boldsymbol{z}')\right] : L_{i+1}^m \to \mathbb{F}_q$ *such that*

$$\mathbf{Fold}\left[f, (\boldsymbol{z}, \boldsymbol{z}')\right](\boldsymbol{y}) = \sum_{\boldsymbol{e}\in\{0,1\}^m} \boldsymbol{z}^{\boldsymbol{e}} g_{\boldsymbol{e}}(\boldsymbol{y}) + \sum_{\substack{\boldsymbol{e}\in\{0,1\}^m \\ \boldsymbol{e}\neq\boldsymbol{0}}} \boldsymbol{z}'^{\boldsymbol{e}} h_{\boldsymbol{e}}(\boldsymbol{y}) g_{\boldsymbol{e}}(\boldsymbol{y}). \tag{4.7}$$

Lemmas 4.23 and 4.24 show that this defines a folding operator for SRM_i as per Definition 3.1.

Lemma 4.23 (Completeness)**.** *Let $(\boldsymbol{z}, \boldsymbol{z}') \in \left(\mathbb{F}_q^m\right)^2$, and $f : L_i^m \to \mathbb{F}_q \in \mathsf{SRM}_i$, then* $\mathbf{Fold}\left[f, (\boldsymbol{z}, \boldsymbol{z}')\right] : L_{i+1}^m \to \mathbb{F}_q$ *belongs to* SRM_{i+1}.

Proof. Proof relies on Proposition 4.6. If $f \in \mathsf{SRM}_i$, then the polynomial $\hat{f}(\boldsymbol{X})$ associated to f has total degree at most $k_i - 1$. Therefore, for any $\boldsymbol{e} \in \{0,1\}^m$, $\deg \hat{g}_{\boldsymbol{e}} \leqslant \left\lfloor \frac{k_i - 1 - \mathsf{w_H}(\boldsymbol{e})}{2} \right\rfloor$. Since k_i is

[1] The introduction of those balancing functions is similar to a technique introduced in [BS08], which is commonly used in the construction of proof systems to handle combinations of polynomials with distinct degree bounds.

even, we have both $\deg \hat{g}_e < k_{i+1}$ and $\deg\left(\hat{h}_e\hat{g}_e\right) \leqslant \left\lfloor \frac{\mathsf{w_H}(e)}{2} \right\rfloor + \left\lfloor \frac{k_i - 1 - \mathsf{w_H}(e)}{2} \right\rfloor < k_{i+1}$. This means $\mathbf{Fold}\left[f,(z,z')\right] : L_{i+1}^m \to \mathbb{F}_q$ corresponds to the evaluation of a polynomial in $\mathbb{F}_q[\boldsymbol{X}]$ of total degree less than k_{i+1}. □

Lemma 4.24 (Locality)**.** *Let $f : L_i^m \to \mathbb{F}_q$ be an arbitrary function and let $(z,z') \in \left(\mathbb{F}_q^m\right)^2$. Given $y \in L_{i+1}^m$, the value $\mathbf{Fold}\left[f,(z,z')\right](y)$ can be computed with exactly 2^m queries to f.*

Proof. The proof follows from the one of Lemma 4.9. For any $y \in L_{i+1}^m$, the vector $(g_e(y))_{e\in\{0,1\}^m}$ corresponds to the vector of coefficients of the low-degree extension of the function $f_{|S_y}$, where

$$S_y \coloneqq \{x \in L_i^m \mid y = (q_i(x_1),\ldots,q_i(x_m))\}$$

and q_i is the polynomial defined in Section 2.1.4. □

Let us now show that the folding operator of Definition 4.22 satisfies distance preservation (Definition 3.4).

Proposition 4.25 (Distance preservation)**.** *Denote λ_{i+1} the minimum relative distance of SRM_{i+1}. Let $f : L_i^m \to \mathbb{F}_q$ be an arbitrary function. Let $\varepsilon \in \left(0, \frac{2}{3}\right)$ and*

$$\delta < \min\left(1 - (1-\lambda_{i+1}+\varepsilon)^{\frac{1}{3}}, \frac{1}{2}(\lambda_{i+1} + m\frac{\varepsilon}{2})\right). \tag{4.8}$$

Let $\varphi_i : L_i^m \to [0,1]$ and $\varphi_{i+1} : L_{i+1}^m \to [0,1]$ be weight functions such that

$$\forall y \in L_{i+1}^m,\ \varphi_{i+1}(y) \leqslant \frac{1}{2^m}\sum_{x\in S_y}\varphi_i(x).$$

If $f : L_i^m \to \mathbb{F}_q$ has weighted agreement $\mathsf{agree}_{\varphi_i}(f,\mathsf{SRM}_i) < 1-\delta$, then

$$\Pr_{z,z'\in\mathbb{F}_q^m}\left[\mathsf{agree}_{\varphi_{i+1}}\left(\mathbf{Fold}\left[f,(z,z')\right],\mathsf{SRM}_{i+1}\right) > 1-\delta+m\varepsilon\right] < \frac{16m}{\varepsilon^2 q}.$$

Proof. Let $f : L_i^m \to \mathbb{F}_q$ be such that $\mathsf{agree}_{\varphi_i}(f,\mathsf{SRM}_i) < 1-\delta$, and $(\hat{g}_e)_{e\in\{0,1\}^m}$ the 2^m m-variate polynomials appearing in the decomposition of $\hat{f}$ in Proposition 4.6. For any $z \in \mathbb{F}_q^m$, denote u_z the function $u_z = \sum_{e\in\{0,1\}^m} z^e g_e$, and for any $e \in \{0,1\}^m \setminus \{\mathbf{0}\}$, define $u_e = h_e g_e$. One can rewrite $\mathbf{Fold}\left[f,(z,z')\right]$ as follows:

$$\mathbf{Fold}\left[f,(z,z')\right] = u_z + \sum_{\substack{e\in\{0,1\}^m \\ e\neq\mathbf{0}}} z'^e u_e.$$

We proceed by contraposition, assuming that

$$\Pr_{z,z'\in\mathbb{F}_q^m}\left[\mathsf{agree}_{\varphi_{i+1}}\left(\mathbf{Fold}\left[f,(z,z')\right],\mathsf{SRM}_{i+1}\right) > 1-\delta+m\varepsilon\right] \geqslant \frac{16m}{\varepsilon^2 q},$$

or, in other words,

$$\Pr_{z\in\mathbb{F}_q^m}\left[\Pr_{z'\in\mathbb{F}_q^m}\left[\mathsf{agree}_{\varphi_{i+1}}\left(\mathbf{Fold}\left[f,(z,z')\right],\mathsf{SRM}_{i+1}\right) > 1-\delta+m\varepsilon\right] \geqslant \frac{8m}{\varepsilon^2 q}\right] \geqslant \frac{8m}{\varepsilon^2 q}.$$

Let

$$A \coloneqq \left\{ \boldsymbol{z} \in \mathbb{F}_q^m \mid \Pr_{\boldsymbol{z}' \in \mathbb{F}_q^m} \left[\mathsf{agree}_{\varphi_{i+1}} \left(\mathbf{Fold} \left[f, (\boldsymbol{z}, \boldsymbol{z}') \right], \mathsf{SRM}_{i+1} \right) > 1 - \delta + m\varepsilon \right] \geqslant \frac{8m}{\varepsilon^2 q} \right\}.$$

Proposition 3.13 implies that, for any $\boldsymbol{z} \in A$, there exist $T_{\boldsymbol{z}} \subset L_{i+1}^m$ and $(w_{\boldsymbol{z},\boldsymbol{e}})_{\boldsymbol{e} \in \{0,1\}^m}$ with $w_{\boldsymbol{z},\boldsymbol{e}} \in \mathsf{SRM}_{i+1}$ such that

- $\sum_{\boldsymbol{y} \in T_{\boldsymbol{z}}} \varphi_{i+1}(\boldsymbol{y}) \geqslant (1 - \delta + m\frac{\varepsilon}{2}) \left| L_{i+1}^m \right|$,
- $w_{\boldsymbol{z},\boldsymbol{0}|T_{\boldsymbol{z}}} = u_{\boldsymbol{z}|T_{\boldsymbol{z}}}$,
- for each $\boldsymbol{e} \in \{0,1\}^m \setminus \{\boldsymbol{0}\}$, $w_{\boldsymbol{z},\boldsymbol{e}|T_{\boldsymbol{z}}} = u_{\boldsymbol{e}|T_{\boldsymbol{z}}}$.

Thus, for all $\boldsymbol{z} \in A$,

$$\mathsf{agree}_{\varphi_{i+1}} \left(\sum_{\boldsymbol{e} \in \{0,1\}^m} \boldsymbol{z}^{\boldsymbol{e}} g_{\boldsymbol{e}}, \mathsf{SRM}_{i+1} \right) \geqslant \frac{1}{\left| L_{i+1}^m \right|} \sum_{\boldsymbol{y} \in T_{\boldsymbol{z}}} \varphi_{i+1}(\boldsymbol{y}) \geqslant 1 - \delta + m\frac{\varepsilon}{2}.$$

Since $|A| > \frac{2m}{\varepsilon^2} q^{m-1}$, we have

$$\Pr_{\boldsymbol{z} \in \mathbb{F}_q^m} \left[\mathsf{agree}_{\varphi_{i+1}} \left(\sum_{\boldsymbol{e} \in \{0,1\}^m} \boldsymbol{z}^{\boldsymbol{e}} g_{\boldsymbol{e}}, \mathsf{SRM}_{i+1} \right) > 1 - \delta + m\frac{\varepsilon}{2} \right] \geqslant \frac{8m}{\varepsilon^2 q}.$$

Again, by Proposition 3.13, we obtain $T \subset L_{i+1}^m$ and $(v_{\boldsymbol{e}})_{\boldsymbol{e} \in \{0,1\}^m}$ with $v_{\boldsymbol{e}} \in \mathsf{SRM}_{i+1}$ such that

- $\sum_{\boldsymbol{y} \in T} \varphi_{i+1}(\boldsymbol{y}) \geqslant (1 - \delta) \left| L_{i+1}^m \right|$,
- for each $\boldsymbol{e} \in \{0,1\}^m$, $v_{\boldsymbol{e}|T} = g_{\boldsymbol{e}|T}$.

Fix $\boldsymbol{z} \in A$. For any $\boldsymbol{e} \in \{0,1\}^m$, $\boldsymbol{e} \neq \boldsymbol{0}$, we have

$$w_{\boldsymbol{z},\boldsymbol{e}|T_{\boldsymbol{z}} \cap T} = u_{\boldsymbol{e}|T_{\boldsymbol{z}} \cap T} = (h_{\boldsymbol{e}} g_{\boldsymbol{e}})_{|T_{\boldsymbol{z}} \cap T} = (h_{\boldsymbol{e}} v_{\boldsymbol{e}})_{|T_{\boldsymbol{z}} \cap T}.$$

Besides, recalling (4.8), the intersection of $T_{\boldsymbol{z}}$ and T satisfies

$$\begin{aligned} |T_{\boldsymbol{z}} \cap T| &= |T_{\boldsymbol{z}}| + |T| - |T_{\boldsymbol{z}} \cup T| \\ &\geqslant \sum_{\boldsymbol{y} \in T_{\boldsymbol{z}}} \varphi_{i+1}(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in T} \varphi_{i+1}(\boldsymbol{y}) - \left| L_{i+1}^m \right| \\ &\geqslant \left(1 - 2\delta + m\frac{\varepsilon}{2} \right) \left| L_{i+1}^m \right|, \\ &\geqslant (1 - \lambda_{i+1}) \left| L_{i+1}^m \right|. \end{aligned}$$

Since λ_{i+1} is the minimum relative distance of SRM_{i+1}, we deduce that $w_{\boldsymbol{z},\boldsymbol{e}} = h_{\boldsymbol{e}} v_{\boldsymbol{e}}$ for every $\boldsymbol{e} \in \{0,1\}^m \setminus \{\boldsymbol{0}\}$.

For any $\boldsymbol{e} \in \{0,1\}^m$, consider polynomials $\hat{v}_{\boldsymbol{e}}, \hat{w}_{\boldsymbol{e},\boldsymbol{z}} \in \mathbb{F}_q[\boldsymbol{X}]$ of total degrees at most k_{i+1}, such that for all $\boldsymbol{x} \in L_{i+1}^m$, $\hat{v}_{\boldsymbol{e}}(\boldsymbol{x}) = v_{\boldsymbol{e}}(\boldsymbol{x})$ and $\hat{w}_{\boldsymbol{e},\boldsymbol{z}}(\boldsymbol{x}) = w_{\boldsymbol{e},\boldsymbol{z}}(\boldsymbol{x})$. Hence, for all $\boldsymbol{x} \in L_{i+1}^m$,

$$\hat{w}_{\boldsymbol{e},\boldsymbol{z}}(\boldsymbol{x}) = \hat{v}_{\boldsymbol{e}}(\boldsymbol{x}) \hat{h}_{\boldsymbol{e}}(\boldsymbol{x}),$$

which means that

$$\hat{w}_{e,z} - \hat{v}_e \hat{h}_e = 0 \bmod \left(Z_{i+1}(X_1), \ldots, Z_{i+1}(X_m)\right), \tag{4.9}$$

where $Z_{i+1}(X) = \prod_{a \in L_{i+1}} (X - a)$ has degree $|L_{i+1}|$. Since $k_{i+1} < |L_{i+1}|$, we have that for any j, $\deg_{X_j} \hat{v}_e \leqslant |L_{i+1}| - 2$. Moreover, $\deg_{X_i} \hat{h}_e \leqslant 1$, thus the above equality is true without the modulo:

$$\hat{w}_{e,z} - \hat{v}_e \hat{h}_e = 0. \tag{4.10}$$

Therefore, $\deg \hat{v}_e < k_{i+1} - \left\lfloor \frac{\mathsf{w_H}(e)}{2} \right\rfloor$. For all $\boldsymbol{e} \in \{0,1\}^m$, we have

$$\deg \boldsymbol{X}^{\boldsymbol{e}} \hat{v}_{\boldsymbol{e}}(q_i(X_1) \ldots, q_i(X_m)) \leqslant \mathsf{w_H}(\boldsymbol{e}) + 2\left(k_{i+1} - 1 - \frac{\mathsf{w_H}(\boldsymbol{e})}{2}\right) < k_i,$$

hence the polynomial $R \in \mathbb{F}_q[\boldsymbol{X}]$ defined by

$$R(\boldsymbol{X}) \coloneqq \sum_{\boldsymbol{e} \in \{0,1\}^m} \boldsymbol{X}^{\boldsymbol{e}} \hat{v}_{\boldsymbol{e}}(q_i(X_1) \ldots, q_i(X_m))$$

has total degree $\deg R < k_i$. Thus the evaluation of R on L_i^m is a codeword $v \in \mathsf{SRM}_i$. For any $\boldsymbol{y} \in T$ and $\boldsymbol{x} \in S_{\boldsymbol{y}}$, we have

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{e} \in \{0,1\}^m} \boldsymbol{x}^{\boldsymbol{e}} g_{\boldsymbol{e}}(\boldsymbol{y}) = \sum_{\boldsymbol{e} \in \{0,1\}^m} \boldsymbol{x}^{\boldsymbol{e}} v_{\boldsymbol{e}}(\boldsymbol{y}) = v(\boldsymbol{x}).$$

Hence, v agrees with the function f on the set $S_T \coloneqq \bigsqcup_{\boldsymbol{y} \in T} S_{\boldsymbol{y}}$. Since $v \in \mathsf{SRM}_i$, we have

$$\mathsf{agree}_{\varphi_i}(f, \mathsf{SRM}_i) \geqslant \frac{1}{|L_i^m|} \sum_{\boldsymbol{x} \in S_T} \varphi_i(\boldsymbol{x}) = \frac{1}{|L_i^m|} \sum_{\boldsymbol{y} \in T} \sum_{\boldsymbol{x} \in S_{\boldsymbol{y}}} \varphi_i(\boldsymbol{x}) \geqslant \frac{1}{|L_{i+1}^m|} \sum_{\boldsymbol{y} \in T} \varphi_{i+1}(\boldsymbol{y}).$$

Eventually, we conclude that $\mathsf{agree}_{\varphi_i}(f, \mathsf{SRM}_i) \geqslant 1 - \delta$ by definition of T, which is a contradiction. □

Remark 4.26. *Note that for Reed-Muller codes whose degree bound is larger than $|L|$, we would not be able to deduce* (4.10) *from* (4.9)*. Nonetheless, for applications to proof systems, the evaluation map is typically injective,* i.e. *$k \leqslant |L|$.*

4.5.3 IOPP for short Reed-Muller codes

Given a sequence of codes $(\mathsf{SRM}_i)_{0 \leqslant i \leqslant r}$ as defined in Section 4.5.1 and a family of folding operators for each code SRM_i (see Section 4.5.2), the generic construction described proposed in Section 3.1.2 leads to a public-coin IOPP $(\mathscr{P}_{\mathsf{RM}}, \mathscr{V}_{\mathsf{RM}})$ for the code SRM_0. As in Section 4.3, the last function f_r is supposed to be constant. Therefore, we use the variant of the protocol described in Remark 3.7. Specifically, instead of sending f_r during the COMMIT phase, the prover $\mathscr{P}_{\mathsf{RM}}$ sends a single field element $\beta \in \mathbb{F}_q$. The verifier $\mathscr{V}_{\mathsf{RM}}$ does not run a membership test to C_r but checks the equation $\beta = \mathbf{Fold}\left[f_{r-1}, \boldsymbol{z}_{r-1}\right](\boldsymbol{y}_r)$. The properties of the resulting IOPP system $(\mathscr{P}_{\mathsf{RM}}, \mathscr{P}_{\mathsf{RM}})$ are displayed in the following theorem.

Theorem 4.27. *Let k, m be positive integers. Assume k is a power of two. Let $L \subset \mathbb{F}_q^\times$ as described in Section 2.1.4 such that $|L| > k$. Then Construction 3.5 with the folding operators defined as per Section 4.5.2 yields a public-coin IOPP system $(\mathscr{P}_{\mathsf{RM}}, \mathscr{V}_{\mathsf{RM}})$ for testing proximity of a function $f : L^m \to \mathbb{F}_q$ to the Short Reed-Muller code $\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$ satisfying*

Completeness: *If $f \in \mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$ and if the oracles $f_1, \dots f_r$ are computed by an honest prover, then $\mathscr{V}_{\mathsf{RM}}$ outputs* ***accept*** *with probability 1.*

Soundness: *Assume that $f : L^m \to \mathbb{F}_q$ is δ-far from $\mathsf{SRM}\left[\mathbb{F}_q, L, m, k\right]$. Denote $\lambda = 1 - \frac{k}{|L|}$. For any $\varepsilon \in \left(0, \frac{2}{3}\right)$, set $\gamma(\varepsilon, \lambda) \coloneqq \min\left(1 - (1 - \lambda + \varepsilon)^{1/3}, \frac{1}{2}(\lambda + m\frac{\varepsilon}{2})\right)$. Then, for any unbounded prover $\mathscr{P}^*$, the verifier $\mathscr{V}_{\mathsf{RM}}$ outputs accept after α repetitions of the QUERY phase with probability at most*

$$\frac{16m\log k}{\varepsilon^2 q} + (1 - \min(\delta, \gamma(\varepsilon, \lambda)) + \varepsilon m \log k)^{\alpha}.$$

Moreover, $(\mathscr{P}_{\mathsf{RM}}, \mathscr{V}_{\mathsf{RM}})$ has the following properties:

- *rounds complexity* $\mathsf{r} < \log k$,
- *proof length* $\mathsf{l} < \frac{n^m}{2^m - 1}$,
- *query complexity* $\mathsf{q} = \alpha 2^m \log k + 1$,
- *prover complexity* $\mathsf{tp} < \left(\frac{11}{2}m + 14\right) n^m$,
- *verifier complexity* $\mathsf{tv} < \alpha 2^m \left(\frac{11}{4}m + 7\right) \log k$.

Proof. We apply the construction of the public-coin IOPP system presented in Section 3.1.2 with the family of folding operators defined in Section 4.5.2. Completeness and soundness follow from Theorem 3.8. The number of rounds is $r = \log k < \log |L|$. Query complexity and proof length are the same as in Theorem 4.11. For soundness, recall that $\min_i \lambda_i \geqslant 1 - \frac{k}{|L|}$ where λ_i is the relative distance of SRM_i.

Let $f : L_i^m \to \mathbb{F}_q$ be an arbitrary function and let $(z, z') \in \left(\mathbb{F}_q^m\right)^2$. We analyze prover complexity by first computing the cost of evaluating $\mathbf{Fold}\left[f, (z, z')\right]$ on L_{i+1}^m. The prover $\mathscr{P}_{\mathsf{RM}}$ can compute the vectors $\left(z^e\right)_{e \in \{0,1\}^m}$ and $\left(z'^e\right)_{e \in \{0,1\}^m}$ in less than $2 \cdot 2^m$ multiplications. Given $y \in L_{i+1}^m$, we look at the cost of computing $\mathbf{Fold}\left[f, (z, z')\right](y)$ (see Equation (4.7)). Recalling Definition 4.21, computing the values $\hat{h}_e(y)$ for all $e \in \{0,1\}^m$ takes at most $m2^{m-2}$ operations. As shown in proof of Lemma 4.9, the vector $(g_e(y))$ corresponds to the coefficients of the multilinear low-degree extension of $f_{|S_y}$. By Lemma 4.3, this interpolation can be performed with $5m2^{m-1}$ arithmetic operations. Prover then computes the first sum of Equation (4.7) using 2^m multiplications and $2^m - 1$ additions. Similarly, the second sum can be computed in less than $3 \cdot 2^m$ arithmetic operations.

Overall, for any function $f : L_i^m \to \mathbb{F}_q$ and $z, z' \in \mathbb{F}_q^m$, the prover can evaluate $\mathbf{Fold}\left[f, (z, z')\right] : L_{i+1}^m \to \mathbb{F}_q$ in less than

$$2 \cdot 2^m + 5 \cdot 2^m \left(1 + \frac{11}{20}m\right) \left|L_{i+1}^m\right| \leqslant 2^m \left(\frac{11}{4}m + 7\right) \left|L_{i+1}^m\right|$$

arithmetic operations. We deduce that the cost of honestly generating $\mathscr{P}_{\mathsf{RS}^m}$'s messages is

$$\sum_{i=0}^{r-1} 2^m \left(\frac{11}{4}m + 7\right) \left|L_{i+1}^m\right| < 2^m \left(\frac{11}{4}m + 7\right) \frac{n^m}{2^m - 1} \leqslant \left(\frac{11}{2}m + 14\right) n^m.$$

From the discussion about prover complexity, we also get that the number of operations made by $\mathscr{V}_{\mathsf{RM}}$ for a single consistency test is less than $2 \cdot 2^m + 5 \cdot 2^m \left(1 + \frac{11}{20}m\right)$. Thus, verifier complexity is less than $\alpha r 2^m \left(\frac{11}{4}m + 7\right)$. □

Comparisons with the univariate case. When we compared the FRI protocol with our IOPP for the tensor product of RS codes in Section 4.3.3, we argued that soundness is affected by the worse relative distance of tensor codes. In constrast, a short Reed-Muller code $\mathsf{SRM}\left[\mathbb{F}_q, L, k, m\right]$ has relative distance which is at least the one of a Reed-Solomon code $\mathsf{RS}\left[\mathbb{F}_q, L, k\right]$. However, soundness of our IOPP for Reed-Muller code is worse than soundness of the FRI protocol for linear-size field [BGKS20] due to the more complex expression of the folding operators.

In Figure 4.3, we present the parameters of the FRI protocol for RS codes and our IOPP for Reed-Muller codes side by side for codes of blocklength N and a single repetition of the QUERY phase. The use of balancing functions in Definition 4.22 and the fact that the partial folding approach cannot be carried over induce some extra costs compared to the IOPP for product codes.

Scheme	Prover	Verifier	Query	Length	Rounds
RS IOPP [BBHR18a]	$< 6N$	$< 42 \log N$	$< 2 \log N$	$< N/3$	$< \log N$
RM IOPP	$< \left(\frac{11}{2}m + 14\right) N$	$2^m \left(\frac{11}{4} + \frac{7}{m}\right) \log N$	$< \frac{2^m}{m} \log N$	$< \frac{N}{2^m - 1}$	$< \frac{\log N}{m}$

Figure 4.3: Comparison between the IOPP for a RS code of [BBHR18a] and our IOPPs for tensor of RS codes and RM codes. Blocklength of the codes is denoted by N and m is the number of variables of the multivariate codes.

Chapter 5

Proximity testing for algebraic geometry codes

5.1 Introduction

5.1.1 Motivations

Algebraic Geometry (AG) codes [Gop77], as evaluations of a set of functions at some designated rational points on a given curve, extend the notion of Reed-Solomon codes. AG codes inherit many of the interesting properties of RS codes while overcoming their main drawback of RS codes, namely requiring an alphabet larger than their length. Therefore, replacing RS codes with AG codes is not only natural but has also led to improvements in the past. Examples of applications of AG codes include public key cryptography, distributed storage, secret sharing and multi-party computation. A key feature for a family of codes to be suitable for arithmetization is a multiplication property [Mei13], namely the component-wise multiplication of two codewords results in codewords in a code whose minimum distance is still good. It should be noted that recent work constructed proof systems based on codes that do not have this feature [RR20, BCG20, BCL20]. The multiplication property actually emulates multiplication of low-degree polynomials. Algebraic geometry codes not only feature this multiplication property but may also have arbitrary large length given a fixed finite field $\mathbb{F}$ (unlike RS codes), with excellent parameters [TVZ82].

Limitations of Reed-Solomon codes. We identify two limitations of using RS codes in IOPs.

As mentioned earlier, RS codes are the simplest case of AG codes, but possess an inherent limitation: the alphabet size must be larger than the block length of the code. Therefore, practical IOP-based succinct arguments are designed over *large fields*.

The second limitation is related to the algebraic structure of the field. RS-IOPPs [BS08, BBHR18a] require the set $D \subset \mathbb{F}$ of evaluation points to have a special structure. Concretely, the field must contain a subgroup of *large smooth order*, typically a power of 2 which is larger than the size of the non-deterministic computation to be verified. Depending on the applications of succinct non-interactive arguments, a base field might already be imposed. This is for instance the case for standard digital signature schemes. For computations whose size exceeds the order of the largest smooth subgroup of the field, RS-IOPPs known to date can no longer be used.

We observe that lowering the size of field elements may not shorten the length of IOP-based succinct non-interactive arguments [BCS16], since the alphabet size is a low-order term in communication complexity. There are, however, other reasonable motivations to replace RS codes with AG ones. We explain below how AG codes could circumvent the limitations of RS codes.

Why can AG codes be useful? First, working over smaller fields lowers the cost of field operations[1]. For concrete efficiency, complexity measures such as prover time and verifier time are closely examined. Reducing significantly the size of the alphabet would have a direct impact on the binary cost of arithmetic operations. Smaller fields could enhance efficiency of proof systems since arithmetization of general circuits would be more efficient. Moreover, on the prover side, the bit complexity of encoding codewords might be smaller.

A popular belief is that encoding with AG codes is an heavy task. It is surely true in general, but there are explicit families of AG codes for which there are quasilinear time encoding algorithms. We discuss more about encoding in Section 5.1.2. On another note, putting forward applications of AG codes can motivates the study of fast coding algorithms for AG codes, in particular in the computer algebra community.

One may be concerned by the overhead of reducing the alphabet size when targeting a soundness error less than $2^{-\kappa}$. Notice that it is possible to sample enough bits of randomness from an extension field when needed, or to repeat only some parts of the protocol (see [Sta21a, BBHR18b]). For instance, soundness error of our IOPP is bounded from below by $|\mathbb{F}|^{-1}$. Reaching the targeting soundness requires to repeat the interactive phase of the IOPP s times, inducing a factor $s \simeq \frac{\kappa}{\log|\mathbb{F}|}$ multiplicative overhead for the prover. A rough estimation of bit complexities does not show evidence of a significant overhead. Overall, a proof system supporting small fields might be more efficient: any part of the protocol which does not contribute to the soundness error could benefit from cheaper field operations.

In addition, AG codes offers more flexibility on the choice of the field. For computations of size n, we propose AG codes for which the field is not required to admit an n-th root of unity (unlike RS-IOPP on a prime field). Specifically, for AG codes over Kummer curves, the base field needs only to have a N-th root of unity, where N divides n. For AG codes over curves in a Hermitian tower (which admits a polylogarithmic-size alphabet), *our IOPP does not involve any assumption on the alphabet*, except that it must be a degree-2 extension of a field $\mathbb{F}_q$, where q is any prime power.

Finally, the question of whether there exist concretely efficient IOPPs for AG codes is motivated by both a theoretical and practical perspective.

5.1.2 Summary of the results

We first give an overview of the contributions of this chapter:

- The first one is to give a clear criterion for constructing IOPPs with linear proof length and sublinear query complexity for AG codes. Our hope with this result is to open up new possibilities for designing efficient probabilistic proof systems based on constant rate AG codes.
- The second contribution is a concrete instantiation for AG codes defined over Kummer-type curves. This IOPP has strictly linear prover time and strictly logarithmic verification (counted in field operations). Thus, we give a strict generalization of the FRI protocol for codes of length n over alphabet of size roughly $n^{2/3}$.

[1]Consider the application of checking the correct execution of a size n computation. Then an RS-based IOP for this problem will work over a field of size $\Omega(n)$. This means that a single addition of two field elements will cost $\Omega(\log n)$ operations. If the IOP is instead based on a code with polylogarithmic-size alphabet, the cost of a single addition is only $\Omega(\log(\text{polylog}(n))$.

– The third one is a concrete instantiation for AG codes defined over a tower of Hermitian curves. Considering recursive towers enables to construct an IOPP for AG codes with *polylogarithmic-size* alphabet. For those codes, we give an IOPP with quasilinear prover time and polylogarithmic verification (counted in field operations).

We recall that complexities are given in field elements and field operations, where the field is the alphabet of the considered code. Asymptotic complexities are relative to the length of the code.

Efficiency of our two AG-IOPP instantiations leverages the fact that proximity testing for these families of AG codes can be reduced to a proximity test for a small RS code.

(1) Generic criterion for constructing AG-IOPPs. Let $\mathcal{X}_0$ be a curve defined over a finite field $\mathbb{F}$, D_0 a divisor on the curve $\mathcal{X}_0$ and $\mathcal{P}_0 \subset \mathcal{X}(\mathbb{F})$. This defines an AG code $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$. We construct a sequence of curves

$$\mathcal{X}_0 \xrightarrow{\pi_0} \mathcal{X}_1 \xrightarrow{\pi_1} \mathcal{X}_2 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_{r-1}} \mathcal{X}_r,$$

so that $\mathcal{X}_{i+1}$ arises as the quotient of the curve $\mathcal{X}_i$ by some automorphism subgroup Γ_i under the quotient map π_i.

Using these consecutive projection maps, we construct a sequence of AG codes $C_i \coloneqq C(\mathcal{X}_i, \mathcal{P}_i, D_i)$ of decreasing length to turn the proximity test of the function $f^{(0)} = f$ to C_0 into a membership test of a function $f^{(r)}$ in C_r. We show that such a procedure is possible if a large (with respect to the length of the code C_0) solvable group $\mathcal{G}$ acts on the initial curve $\mathcal{X}_0$, and under some hypotheses on the divisor D_0 overviewed in Section 5.1.3 and detailed in Section 5.3. An AG code fulfilling all the required conditions is called *foldable*.

Assuming that an AG code $C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ of blocklength n is foldable, we show that there is an $O(\log n)$-rounds IOPP for it, with linear proof length, sublinear query complexity and constant soundness (see Theorem 5.25).

In general, we observe that the larger is the group $\mathcal{G}$ acting on $\mathcal{X}_0$ compared to n, the smaller are the query complexity and the verifier decision complexity of the protocol.

However, we notice that the hypothesis on the size of $\mathcal{G}$ is not a necessary condition for constructing an IOPP with sublinear verification. For instance, if the curve $\mathcal{X}_r$ is isomorphic to the projective line $\mathbb{P}^1$, we can continue to recurse in order to reduce even more the size of the proximity testing problem. We propose two interesting families of AG codes for which it is the case.

(2) Concrete IOPP for AG codes on Kummer curves. When $\mathcal{X}$ is a Kummer curve defined by $Y^N = f(X)$, we show how to choose $\mathcal{P}$ and D to make the AG code $C = C(\mathcal{X}, \mathcal{P}, D)$ foldable. We benefit from the action of the group $\mathbb{Z}/N\mathbb{Z}$ on $\mathcal{X}$ that yields a quotient curve $\mathcal{X}/(\mathbb{Z}/N\mathbb{Z})$ isomorphic to the *projective line*. This enables us to define a sequence of codes $(C_i)_{0 \leqslant i \leqslant r}$ such that $C_0 = C$ and the code C_r is a *Reed-Solomon code* of dimension $(\deg D)/N + 1$.

Theorem 5.1 (Informal, see Theorem 5.51)**.** *Let $C = C(\mathcal{X}, \mathcal{P}, D) \subset \mathbb{F}^{\mathcal{P}}$ be a foldable AG code defined over a Kummer curve $\mathcal{X}$ of equation $\mathcal{X} : Y^N = f(X)$ such that $\deg f = N\ell - 1$ for some integer $\ell > 0$ and N is a smooth integer, coprime with $|\mathbb{F}|$. Assume $\mathbb{F}$ contains a primitive N-th root of unity. The block length $n \coloneqq |\mathcal{P}|$ is a multiple of N and satisfies $n < \ell N^2 |\mathbb{F}|^{1/2}$. Let $f : \mathcal{P} \to \mathbb{F}$ be a purported codeword. For every proximity parameter $\delta \in (0,1)$ and soundness $\varepsilon \in (0,1)$, there*

exists a public-coin IOPP system $(\mathscr{P}, \mathscr{V})$ *for* C *with perfect completeness and the following properties:*

- *rounds complexity* $\mathsf{r}(n) < \log n$,
- *proof length* $\mathsf{l}(n) = O(n)$,
- *query complexity* $\mathsf{q}(n) = O(\log n)$,
- *prover complexity* $\mathsf{tp}(n) = O(n)$,
- *verifier complexity* $\mathsf{tv}(n) = O(\log n)$.

It is worth noting that the Hermitian curve defined over $\mathbb{F}_{q^2}$ by $Y^{q+1} = X^q + X$ satisfies the hypotheses of the previous theorem. It is well known to be *maximal*, i.e. it has the maximum number of rational points with respect to its geometry. We recall that Hermitian codes over alphabet Σ support block length up to $|\Sigma|^{3/2}$, which is greater by a factor $n^{1/3}$ than Reed-Solomon codes.

(3) Concrete IOPP for AG codes on towers of Hermitian curves. We recall that a tower of curves consists of an infinite sequence of curves

$$\mathcal{X}_0 \leftarrow \mathcal{X}_1 \leftarrow \ldots \leftarrow \mathcal{X}_n \leftarrow \ldots$$

such that the number of rational points of the n^{th} curve tends to infinity as n tends to infinity. Towers of curves play a prominent role in the history of AG codes as they define codes with outstanding length and correction capacity [TVZ82, BBGS14].

The Hermitian tower (Definition 5.8) is an example of the widely studied Artin-Scheier extensions [Lac92, Sti09]. In this case, the curve $\mathcal{X}_i$ arises as the quotient of the curve $\mathcal{X}_{i+1}$ modulo the action of a group of order q of the finite field $\mathbb{F}_{q^2}$, the first curve $\mathcal{X}_0$ being isomorphic to the projective $\mathbb{P}^1$. Therefore, one can test proximity to an AG-code from one of the curves $\mathcal{X}_n$ by going down along the tower and then testing proximity to a RS code, whose degree can be expressed explicitly in terms of the initial AG code.

Beyond supporting polylogarithmic-size alphabet, AG codes over the Hermitian tower happen to be more naturally "foldable". In particular, no additional assumptions on the alphabet are required.

We write $\mathrm{polylog}(n)$ for functions that are in $O\left(\log^k(n)\right)$ for some k.

Theorem 5.2 (Informal, see Theorem 5.54)**.** *Let* $C = C(\mathcal{X}, \mathcal{P}, D) \subset \mathbb{F}^{\mathcal{P}}$ *be a foldable AG code over an alphabet* $\mathbb{F}$ *of size* $|\mathbb{F}| = \Omega(\log^k(n))$ *for some constant* k*. We denote* $n = |\mathcal{P}|$*. Let* $f : \mathcal{P} \to \mathbb{F}$ *be a purported codeword. For every proximity parameter* $\delta \in (0,1)$ *and soundness* $\varepsilon \in (0,1)$*, there exists a public-coin IOPP system* $(\mathscr{P}, \mathscr{V})$ *for* C *with perfect completeness and the following properties:*

- *rounds complexity* $\mathsf{r}(n) < \log n$,
- *proof length* $\mathsf{l}(n) = O(n)$,
- *query complexity* $\mathsf{q}(n) = \mathrm{polylog}(n)$,
- *prover complexity* $\mathsf{tp}(n) = \widetilde{O}(n)$,
- *verifier complexity* $\mathsf{tv}(n) = \mathrm{polylog}(n)$.

More on the practicality of AG codes. When constructing a proximity test for a code, it is assumed that the purported codeword is given as input to the prover. Thus, the prover complexity is computed thereof. While we heavily rely on the group of automorphisms of the curve for proving the existence of an efficient IOPP for "foldable" AG codes, we emphasize that the work of the prover and the verifier during the protocol is essentially to perform some univariate polynomial interpolation tasks, with

very small degree. In particular, neither the prover nor the verifier of the IOPP system need to run an encoding algorithm for AG codes.

However, keeping applications to code-based IOP constructions in mind, the running time of the IOP prover is bounded from below by the time needed to encode codewords during arithmetization. Fast encoding algorithms for AG codes is not the most widely studied computational task, and is often a concern when suggesting constructions based on AG codes[2].

This is a reason why we focus our study on families of AG codes that are particularly likely to lead to practical implementations, as we argue next. Specifically, our study includes the two following subfamilies of one-point AG codes over small alphabets with constant rate and distance.

- The first family includes one-point AG codes over Kummer-type curves, and in particular the notorious Hermitian curve. [BRS20] proposed an encoding algorithm with quasilinear complexity $\widetilde{O}(n)$. Roughly speaking, [BRS20] method consists in translating the encoding task into a bivariate polynomial multipoint-evaluation problem. Assuming that the evaluation points are well-structured, they view a bivariate polynomial in $\mathbb{F}[X,Y]$ as a polynomial in $\mathbb{F}[X][Y]$ in order to evaluate it thanks to two univariate multipoint evaluations. It is the same idea than the one for computing m-dimensional FFT from m (univariate) FFTs.
- The second family of one-point AG codes arises from curves on the Hermitian tower and has an alphabet size polylogarithmic in the block length of the code. It is likely that those codes could also be encoded in quasilinear time, by iteratively applying the encoding method proposed by [BRS20].

We also point out that bases for Riemann-Roch spaces related to theses codes are explicitly known.

5.1.3 Overview of our approach

We now present a brief overview of our techniques to generalize the FRI protocol to the AG context. Familiarity with the general ideas used in the FRI protocol is beneficial in following the incoming discussions (see Section 2.1.2).

Group actions and Riemann-Roch spaces. The FRI protocol relies on a decomposition of univariate polynomials of the form

$$f(x) = g_0(\pi(x)) + x g_1(\pi(x)), \tag{5.1}$$

where π is some monic polynomial of degree 2 and g_0, g_1 are polynomials of degree at most $(\deg f)/2$.

Our generalization of the FRI protocol for RS codes $\mathsf{RS}[\mathbb{F}, \mathcal{P}, k]$ to the AG setting comes from the following observation. One can view the splitting of a polynomial $f \in \mathbb{F}[X]$ into an even and an odd parts as coming from the action of a multiplicative group isomorphic to $\mathbb{Z}/2\mathbb{Z}$ on the evaluation set $\mathcal{P}$. This observation is also true when the evaluation domain $\mathcal{P}$ is an additive subgroup of $\mathbb{F}$.

Let $\mathcal{X}$ be a curve defined over a finite field $\mathbb{F}$ and $C = C(\mathcal{X}, \mathcal{P}, D)$ be an AG code. As soon as a group Γ of order p acts on the curve $\mathcal{X}$, its action naturally extends on the functions on $\mathcal{X}$. Let us

[2] In general, the asymptotic cost of the task of encoding an arbitrary linear code of length n is $O(n^2)$ (using a generator matrix for the code).

denote by π the canonical projection $\pi : \mathcal{X} \to \mathcal{X}/\Gamma$. In order to mimic the decomposition (5.1), we wish to write a function of $L_{\mathcal{X}}(D)$ as

$$f = \sum_{j=0}^{p-1} \mu^j f_j \circ \pi \text{ with } f_j \in L_{\mathcal{X}/\Gamma}(E_j), \tag{$\star$}$$

for some function μ on the curve $\mathcal{X}$ and some divisors E_j on the quotient curve that are explicitly expressed in terms of the divisor D. Now assume that no point of $\mathcal{P}$ is fixed by Γ and $\mathcal{P}' = \pi(\mathcal{P})$. Polynomial interpolation enables the determination of $f_j(P)$ for any point $P \in \mathcal{P}'$ with exactly p values of f, namely on the set $\pi^{-1}(\{P\})$. This means that the decomposition ($\star$) can be written for any function in $\mathbb{F}^{\mathcal{P}}$, not only for elements of $L_{\mathcal{X}}(D)$.

Finding a decomposition ($\star$). In the case where $\Gamma = \langle \gamma \rangle$ is a cyclic group whose order is coprime with the characteristic, such a decomposition is guaranteed to exist. Indeed, in this case, a result of Kani [Kan86] states that there exists a function μ on $\mathcal{X}$ satisfying ($\star$) such that $\gamma \cdot \mu = \zeta\mu$ where ζ is a primitive root of unity of order $|\Gamma|$. Moreover, each divisor $E_j, j \in [\![0, p-1]\!]$, can be explicitly written in terms of the divisor D and the function μ.

We may also be able to exhibit such a decomposition without invoking Kani's theorem when Γ is not a cyclic group, just by knowing a basis of $L_{\mathcal{X}}(D)$. This is exactly the strategy we use to design an IOPP for AG codes along the Hermitian tower.

On the existence of distance-preserving folding operators. Our goal is to define a family of folding operators $(\mathbf{Fold}[\cdot, z])_{z \in \mathbb{F}}$ from $\mathbb{F}^{\mathcal{P}}$ to $\mathbb{F}^{\mathcal{P}'}$ and a code $C' = C(\mathcal{X}/\Gamma, \mathcal{P}', D')$ such that

$$\mathbf{Fold}[\cdot, z](C) \subseteq C'.$$

A natural idea is to define the folding operators as a random linear combination of the functions f_j's coming from the decomposition ($\star$) of $f \in \mathbb{F}^{\mathcal{P}}$. Alternatively, we can reduce verifier randomness using standard derandomization techniques, setting for instance

$$\mathbf{Fold}[f, z] = \sum_{j=0}^{p-1} z^j f_j$$

for any $z \in \mathbb{F}$. With this definition, the code C' has to be associated to a divisor D' on $\mathcal{X}/\Gamma$ such that each Riemann-Roch space $L_{\mathcal{X}/\Gamma}(E_j)$ can be embedded into $L_{\mathcal{X}/\Gamma}(D')$. This would indeed define a family of folding operators for the code C' as per Definition 3.1, namely a codeword of C would be mapped to a codeword of C' and the evaluation of $\mathbf{Fold}[f, z]$ would satisfy local computability. However, distance preservation of the folding operators would not be guaranteed, as we discuss next.

The best scenario is when the divisor D yields a decomposition of $L_{\mathcal{X}}(D)$ as p "copies" of the same Riemann-Roch space, as it is the case with Reed-Solomon codes of dimension a power of 2. Unfortunately, to the best of our knowledge, it is unlikely that all divisors E_j involved in the decomposition ($\star$) of f are the same (or even linearly equivalent) *if $\mathcal{X}$ is not the projective line*. We are then facing an issue analogous to the one encountered for Short Reed-Muller codes in Section 4.5.

Similarly to the Reed-Muller case, we wish to define some balancing functions ν_j such that, for every $f_j \in C'$, if the product $\nu_j f_j$ also lies in C', then the function f_j belongs to the desired Riemann-Roch space $L_{\mathcal{X}/\Gamma}(E_j)$. Defining such a balancing function ν_j is tantamount to specify its pole order

at the points supporting the divisor D'. The existence of all the functions ν_j thus depends on the Weierstrass semigroup of these points (see Section 5.2.3 for a definition) and *does not hold for any divisor* D'.

If such functions exist for a divisor D', we will say that D' is *compatible* with D. Finding a convenient divisor D' compatible with a given divisor D is definitely the trickiest part in defining the folding operators properly. This contrasts with the case of polynomial codes, where we are always able to define balancing functions as evaluations of a monomials of appropriate degree.

Once we have a divisor D' that is D-compatible, we shall embed additional terms in the folding operators to account for the balancing functions. We modestly increase the randomness used by the verifier and define distance-preserving folding operators (Definitions 3.1 and 3.4) as follows. For $(z_1, z_2) \in \mathbb{F}^2$, we set

$$\textbf{Fold}\,[f,(z_1,z_2)] = \sum_{j=0}^{p-1} z_1^j f_j + \sum_{j=0}^{p-1} z_2^{j+1} \nu_j f_j.$$

Preserving non trivial soundness of the IOPP. The soundness of Theorem 3.8 depends on the relative minimum distance of the codes of the sequence used in the IOPP. Therefore, distance preservation of the folding operators is not enough to ensure a non trivial soundness of the IOPP protocol when the folding operation is iterated over several rounds. Ideally, we would like the rates of the codes C and C' to be roughly equal to prevent the relative minimum distance from dropping. In other words, we need $L_{\mathcal{X}/\Gamma}(D')$ to be not too large with respect to the components $L_{\mathcal{X}/\Gamma}(E_j)$.

A natural idea would be to choose D' as the divisor E_j with the largest Riemann-Roch space. However, balancing functions only exist for some well-chosen divisors D', whose degree can be significantly larger than the degree of the divisor E_j in $(\star)$. Therefore, the divisors D and D' have to be carefully chosen to also prevent the minimum distance from collapsing.

Defining a sequence of "foldable" AG codes. In view of the above-mentioned points, it seems possible to build an IOPP only for specific families of AG codes. We develop a framework where this is achievable and show that interesting code families fit into it.

With the goal of iterating the folding process in mind, we assume that the base curve $\mathcal{X}_0$ is endowed with a *suitable* acting group $\mathcal{G}$ that we decompose into smaller groups $\Gamma_0, \Gamma_1, \ldots, \Gamma_{r-1}$ to fragment its action and create intermediary quotients

$$\mathcal{X}_0 \xrightarrow{\pi_0} \mathcal{X}_1 \xrightarrow{\pi_1} \mathcal{X}_2 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_{r-1}} \mathcal{X}_r,$$

where the morphism $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ is the quotient map by Γ_i. If $\mathcal{G}$ is a *solvable group*, then (by definition) there is a suitable finite sequence $\Gamma_0, \Gamma_1, \ldots, \Gamma_{r-1}$ that allows to define those intermediary quotients.

In Section 5.3, we will say that a code $C = C(\mathcal{X}, \mathcal{P}, D)$ is a *foldable AG code* if we are able to construct a sequence of AG codes $(C_i)_{0 \leqslant i \leqslant r}$ where each code $C_i \coloneqq C(\mathcal{X}_i, \mathcal{P}_i, D_i)$ supports a family of randomized folding operators $\textbf{Fold}\,[\cdot, z] : \mathbb{F}^{\mathcal{P}_i} \to \mathbb{F}^{\mathcal{P}_{i+1}}$ satisfying Definitions 3.1 and 3.4.

To ensure that the last code C_r has sufficiently small length and to obtain an IOPP with sublinear query complexity, we ask for the size of $\mathcal{G}$ to be greater than $|\mathcal{P}|^e$ for a certain $e \in (0,1)$. We point out that this requirement allows us to give a generic setting which guarantees the existence of an IOPP with sublinear verification time. However, depending on $\mathcal{X}$, the requirement on the size of $\mathcal{G}$

may be not necessary. In fact, in our concrete instantiations of foldable AG codes (Sections 5.5 and 5.6), we manage to obtain logarithmic verification without requiring this condition on $|\mathcal{G}|$.

5.1.4 Related work

We discuss works related to AG-based proximity testing. We emphasize that the motivation behind existing works was only theoretical. In particular, the PCP techniques used are too complex to be implemented for verifying meaningful computations.

In 2013, [BKK+13] constructed a PCP with linear proof length and sublinear query complexity for boolean circuit satisfiability by relying on AG codes. More precisely, for any $\varepsilon > 0$ and instances of size n, their PCP has length $2^{O(1/\varepsilon)}n$ and query complexity n^{ε}. When aiming at optimal proof length and query complexity as small as possible, this result remains the state-of-the-art PCP construction. By using AG codes, the authors of [BKK+13] reduced the field size to a constant, which avoids a logarithmic blowup in proof bit-length (occuring e.g. in [BS08] when using univariate polynomials of degree m to encode binary strings of length m). In [BKK+13], the authors pointed out that they are not able to apply proof composition [AS92] to reduce the query complexity of their PCP because decision complexity of the PCP verifier is too large (polynomial in the query complexity).

Improving on [BKK+13], [BCG+17] proposed an IOP for boolean circuit satisfiability with linear proof length and constant query complexity. The IOP of [BCG+17] invoked the sumcheck protocol [LFKN90] on a $O(1)$-wise tensor product of AG codes (recall that the 2-wise tensor product operation squares the rate and the relative distance). Then, they use Mie's PCP of Proximity for non-deterministic languages [Mie09] to test proximity to the tensored code. Both constructions benefit from the use of AG codes to get constant size alphabet and linear proof bit-lengths.

A recent work of [RR20] constructed an IOPP for any deterministic language which can be decided in time $\text{poly}(n)$ and space $n^{o(1)}$. In particular, [RR20, Corollary 3.6] can be applied to test proximity to AG codes. This IOPP outperforms our construction on some parameters: it has constant round and query complexities, and proof length is slightly less than n. However, it is unlikely that [RR20]'s IOPP leads to a concrete implementation, which is a motivation for our work. Indeed, prover running time is polynomial, and the inner IOPP used for achieving constant query complexity via proof composition is the heavy PCPP of [Mie09]. Mie's PCPP is a theoretical and complex tool used to achieve constant query (e.g in [BCG+17, BCG+19, BCL20]), but it is seen as impractical[3].

By contrast, we exhibit two explicit families of AG codes for which we are able to construct a proximity test with linear prover running time and logarithmic verification (for the first one) and quasilinear prover time with polylogarithmic verification (for the second one). The main point however, is that our construction is undoubtedly much simpler to implement: the most complex task of the prover and the verifier is simply to perform univariate interpolations (with very small degrees). The technical difficulties are in analyzing the conditions allowing the construction, but the protocol itself is very similar to the FRI protocol. IOPP inspired by the FRI protocol have the inherent barrier of logarithmic query complexity. However, in practice, it is still the most efficient proximity test for Reed-Solomon codes known to date.

[3]Proposing an alternative to [Mie09] which does not involve heavy PCP machinery would allow to narrow the gap between the best constructions known in theory and the most efficient ones used in practice.

5.2 Algebraic Geometry Codes

Algebraic-geometry codes are defined by evaluations of rational functions of bounded "order" on a set of points on a "nice" algebraic curve. In this section, we gather the notions and properties that will allow us to define them.

Note that this section serves more as a reminder of basic notions and notations than as a precise introduction to the area. We refer to [Mor91, TVN07, Ful08, Sti09] for exact definitions, proofs and further details on algebraic geometry and algebraic function fields.

We denote by $\mathbb{F}$ a finite field and by $\bar{\mathbb{F}}$ a fixed algebraic closure of $\mathbb{F}$.

5.2.1 Basic notions on algebraic curves over finite fields

Given a field $\mathbb{F}$, we denote by $\mathbb{P}^n(\mathbb{F})$ (or simply $\mathbb{P}^n$) the projective n-space over $\mathbb{F}$, that is the quotient set $\mathbb{P}^n(\mathbb{F}) \coloneqq \left(\mathbb{F}^{n+1} \setminus \{\mathbf{0}\}\right) \big/ \sim$, where $\boldsymbol{a} \sim \boldsymbol{b} \iff \exists \lambda \in \mathbb{F} \setminus \{0\}, \boldsymbol{a} = \lambda \boldsymbol{b}$. The equivalence class of $(x_1, \ldots, x_{n+1})$ will be denoted by $[x_1 : \cdots : x_{n+1}] \in \mathbb{P}^n(\mathbb{F})$. As usual, we say that an element $[x_1 : \cdots : x_{n+1}] \in \mathbb{P}^n(\mathbb{F})$ is a point with projective coordinates $x_1, \ldots, x_{n+1}$.

Affine points of the n-dimensional affine space $\mathbb{A}^n(\mathbb{F})$ can be embedded into $\mathbb{P}^n(\mathbb{F})$ via the map $i : \mathbb{A}^n(\mathbb{F}) \hookrightarrow \mathbb{P}^n(\mathbb{F})$ defined by $i(x_1, \ldots, x_n) = [x_1 : \cdots : x_n : 1]$. We call point at infinity an element in the complement of the image of i, namely a point such that $x_{n+1} = 0$.

Moreover, we refer to the set $\mathbb{P}^1(\mathbb{F}) = \{[x : 1] \mid x \in \mathbb{F}\} \cup \{[1 : 0]\}$ as the projective line over $\mathbb{F}$.

Function fields and dimension of varieties. Recall that a polynomial $f \in \mathbb{F}[\boldsymbol{X}]$ is homogeneous if its monomials have exactly the same degree. The zero locus of a homogeneous polynomial being well-defined, we can consider the following object. A set $\mathcal{V} \subset \mathbb{P}^n(\bar{\mathbb{F}})$ is a *projective algebraic set* if there exists a set of homogeneous polynomials $S \subseteq \bar{\mathbb{F}}[X_1, \ldots, X_{n+1}]$ such that $\mathcal{V} = \left\{P \in \mathbb{P}^n(\bar{\mathbb{F}}) \mid \forall f \in S, f(P) = 0\right\}$. We denote by $I(\mathcal{V})$ the ideal of a projective algebraic set $\mathcal{V}$, namely the ideal generated by the homogeneous polynomials $f \in \bar{\mathbb{F}}[\boldsymbol{X}]$ such that, for all $P \in \mathcal{V}$, $f(P) = 0$. An algebraic set $\mathcal{V}$ is *irreducible* in $\mathbb{F}$ if it cannot be written as the union of two proper algebraic subsets in $\mathbb{F}$. If $\mathcal{V}$ is irreducible in $\bar{\mathbb{F}}$, we say that $\mathcal{V}$ is *absolutely irreducible*.

A *projective variety* $\mathcal{V}$ is an irreducible projective algebraic set. Thus, the ideal of a nonempty projective variety is a prime ideal. For our applications, we shall consider projective varieties $\mathcal{V} \subseteq \mathbb{P}^n(\bar{\mathbb{F}})$ *defined over* $\mathbb{F}$ (we may write $\mathcal{V}/\mathbb{F}$ for short). We define them as projective varieties whose ideal $I(\mathcal{V})$ is generated by homogeneous polynomials with coefficients in $\mathbb{F}$.

Let $\mathcal{V}$ be a nonempty projective variety defined over $\mathbb{F}$. Its coordinate ring $\mathbb{F}[\mathcal{V}] \coloneqq \mathbb{F}[\boldsymbol{X}] \big/ I(\mathcal{V})$ is an integral domain. In order to define rational functions on $\mathcal{V}$, we consider the *function field* of $\mathcal{V}$, defined as

$$\mathbb{F}(\mathcal{V}) \coloneqq \left\{ \frac{U}{V} \in \mathrm{Frac}\left(\mathbb{F}[\mathcal{V}]\right) \mid U, V \text{ are homogeneous polynomials with } \deg U = \deg V \right\}.$$

Let $P \in \mathcal{V}$, $f \in \mathbb{F}(\mathcal{V})$. A function $f \in \mathbb{F}(\mathcal{V})$ is *defined* at $P \in \mathcal{V}$ (or *regular* at P) if there is $\frac{U}{V} \in \mathbb{F}(\mathcal{V})$ such that $f = \frac{U}{V}$ and $V(P) \neq 0$. In that case, the value of f at P is well-defined and equals $U(P)/V(P)$. The set of functions in $\mathbb{F}(\mathcal{V})$ that are defined at P forms a local ring, denoted $\mathcal{O}_P(\mathcal{V})$. Therefore $\mathcal{O}_P(\mathcal{V})$ has a unique maximal ideal, which is $M_P(\mathcal{V}) \coloneqq \{f \in \mathcal{O}_P(\mathcal{V}) \mid f(P) = 0\}$.

The function field of a projective variety $\mathcal{V}$ is a finitely generated extension of $\mathbb{F}$. The number of independent rational functions on a variety $\mathcal{V}$ enables to define the dimension of $\mathcal{V}$. Specifically, the *dimension* of a projective variety $\mathcal{V}$ over $\mathbb{F}$ is the transcendence degree of $\mathbb{F}(\mathcal{V})$ over $\mathbb{F}$.

Projective curves. We will consider AG codes defined over a *projective curve* $\mathcal{X}/\mathbb{F}$, i.e. a projective variety of dimension one. We denote by $\mathcal{X}(\mathbb{F})$ the set of $\mathbb{F}$-rational points of $\mathcal{X}$, namely points of $\mathcal{X}$ that are in $\mathbb{P}^n(\mathbb{F})$.

Let $\mathcal{X}$ be a projective curve with homogeneous ideal $I(\mathcal{X}) = \langle f_1, \ldots, f_r \rangle$. We say that a point P of $\mathcal{X}$ is *singular* if the rank of the Jacobian matrix of the first-order partial derivatives of $f_1, \ldots, f_r$ at P is strictly less than $n-1$, and *non-singular* otherwise.[4] The projective curve $\mathcal{X}/\mathbb{F}$ is said to be *smooth* (or *non-singular*) if every point of $\mathcal{X}$ is a non-singular point.

Let $\mathcal{X}$ be a smooth, projective, absolutely irreducible curve defined over $\mathbb{F}$. Assuming that $\mathcal{X}$ is plane (i.e. contained in $\mathbb{P}^2(\bar{\mathbb{F}})$), it can be defined by a polynomial $f(X,Y) \in \mathbb{F}[X,Y]$ irreducible in $\bar{\mathbb{F}}[X,Y]$. The affine part of $\mathcal{X}$ is the set of points $P \in \bar{\mathbb{F}}^2$ such that $f(P) = 0$. The entire curve $\mathcal{X}$ can be retrieved by considering the zeroes in $\mathbb{P}^2(\bar{\mathbb{F}})$ of the polynomial $f^* \in \mathbb{F}[X,Y,Z]$ obtained by homogenization of $f \in \mathbb{F}[X,Y]$. Since the affine part of such plane projective curves suffices to describe them entirely, we may define them by simply giving the corresponding affine equation. We call degree of such a plane curve $\mathcal{X}$ the degree of f (which is obviously the same as the degree of f^*).

Example 5.3 (Hermitian curve)**.** Suppose $\mathbb{F} = \mathbb{F}_{q^2}$. The Hermitian curve $\mathcal{H}/\mathbb{F}_{q^2}$ is the curve defined as the zero set of the polynomial $Y^qZ + YZ^q - X^{q+1}$. The curve $\mathcal{H}$ is a nonsingular projective plane curve of degree $d = q+1$. The q^3+1 rational points of the Hermitian curves are

$$\mathcal{H}(\mathbb{F}_{q^2}) = \left\{ (x:y:1) \in \mathbb{P}^2(\mathbb{F}_{q^2}) \mid x^{q+1} = y^q + y \right\} \cup \{[0:1:0]\}.$$

Valuations. Let $\mathcal{X}$ be a projective curve and P a non-singular point of $\mathcal{X}$. The local ring

$$\mathcal{O}_P = \{ f \in \mathbb{F}(\mathcal{X}) \mid f \text{ is defined at } P \}$$

is a discrete valuation ring of $\mathbb{F}(\mathcal{X})$. In particular, the unique maximal ideal M_P of $\mathcal{O}_P$ is principal. A function t such that $M_P = t\mathcal{O}_P$ is called a *local parameter* for P. Writing $M_P = t\mathcal{O}_P$, any non-zero element $f \in \mathbb{F}(\mathcal{X})$ has a unique representation $f = t^n u$, where $n \in \mathbb{Z}$, $u \in {\mathcal{O}_P}^\times$ and ${\mathcal{O}_P}^\times$ is the set of units of $\mathcal{O}_P$. Then, we can define a discrete valuation at P as a function $v_P : \mathbb{F}(\mathcal{X}) \to \mathbb{Z} \cup \{\infty\}$ as follows: for any non-zero $f \in \mathbb{F}(\mathcal{X})$, $f = t^n u$, we set $v_P(f) := n$. By convention, we define $v_P(0) = \infty$. We say that $v_P(f)$ is the *valuation* of $f \in \mathbb{F}(\mathcal{X})$ at P. A point P is said to be a *zero* of f if $v_P(f) > 0$, and a *pole* of f if $v_P(f) < 0$. Any non-zero function of $\mathbb{F}(\mathcal{X})$ has a finite number of zeros and poles.

Divisors. Let $\mathcal{X}/\mathbb{F}$ be a curve. A divisor D on $\mathcal{X}$ is a formal sum of points $D := \sum_{P \in \mathcal{X}} n_P P$, with coefficients $n_P \in \mathbb{Z}$ which are all zero, except for a finite number of them. The support of D, denoted $\mathrm{supp}(D)$, is the finite set of points P such that n_P is non zero. For our applications, we will consider only divisors supported by $\mathbb{F}$-rational points. The degree of a rational divisor $D = \sum n_P P$

[4] Since $P \in \mathbb{P}^n$, the entries $\frac{\partial f_i}{\partial x_j}(P)$ of the Jacobian matrix are not well-defined. However, multiplying the coordinates of P by a scalar λ will multiply $\frac{\partial f_i}{\partial x_j}(P)$ by $\lambda^{\deg f_i - 1}$, and row transformations does not affect the rank of the matrix.

is $\deg D \coloneqq \sum n_P$. We say that a divisor D is *effective* if $n_P \geqslant 0$ for every point P. We denote by $\mathrm{Div}(\mathcal{X})$ the set of divisors on $\mathcal{X}$. It is endowed with a partial order relation $\leqslant$ such that $D \leqslant D'$ if $D' - D$ is effective.

Let $\varphi : \mathcal{X} \to \mathcal{X}'$ be a map between two algebraic curves $\mathcal{X}$ and $\mathcal{X}'$. For $f \in \mathbb{F}(\mathcal{X}')$, we denote by φ^* the map $\varphi^* : \mathbb{F}(\mathcal{X}') \to \mathbb{F}(\mathcal{X})$ defined by $\varphi^* f \coloneqq f \circ \varphi$, and refer to it as the *pull-back* map induced by φ. Given a divisor $D = \sum_P n_P P$ on $\mathcal{X}$, the *push-forward* of D, denoted $\pi_*(D)$, is the divisor on $\mathcal{X}'$ defined by $\pi_*(D) \coloneqq \sum_P n_P \varphi(P)$.

An element f of the function field $F \coloneqq \mathbb{F}(\mathcal{X})$ of the curve $\mathcal{X}$ defines a principal divisor

$$\mathrm{div}_{\mathcal{X}}(f) \coloneqq \sum_{P \in \mathcal{X}} v_P(f) P,$$

where v_P is the valuation of f at P. The subscript $\mathcal{X}$ will be omitted when the context is clear.

We denote by $\mathrm{div}_0(f)$ (respectively $\mathrm{div}_\infty(f)$) the positive (respectively negative) part of the principal divisor $\mathrm{div}(f)$, *i.e.*

$$\mathrm{div}_0(f) \coloneqq \sum_{\substack{P \in \mathcal{X} \\ v_P(f) > 0}} v_P(f) P \quad \text{and} \quad \mathrm{div}_\infty(f) \coloneqq \sum_{\substack{P \in \mathcal{X} \\ v_P(f) < 0}} v_P(f) P,$$

so that $\mathrm{div}(f) = \mathrm{div}_0(f) - \mathrm{div}_\infty(f)$. The divisors $\mathrm{div}_0(f)$ and $\mathrm{div}_\infty(f)$ correspond to the loci of zeroes and poles of f, respectively.

Two divisors $D_1, D_2 \in \mathrm{Div}(\mathcal{X})$ are *linearly equivalent* if there is a non-zero function $f \in \mathbb{F}(\mathcal{X})$ such that $D_1 = D_2 + \mathrm{div}(f)$.

Riemann-Roch spaces. The *Riemann-Roch space* of a divisor $D \in \mathrm{Div}(\mathcal{X})$ is the finite-dimensional vector space over $\mathbb{F}$ defined by

$$L_{\mathcal{X}}(D) \coloneqq \{f \in \mathbb{F}(\mathcal{X}) \mid \mathrm{div}(f) + D \geqslant 0\} \cup \{0\}.$$

In particular, a function $f \in L_{\mathcal{X}}(D)$ has no pole outside of $\mathrm{supp}(D)$.

The subscript specifying the curve $\mathcal{X}$ will be omitted when it is clear from the context. Recall that we have $L_{\mathcal{X}}(D) \subseteq L_{\mathcal{X}}(D')$ whenever $D \leqslant D'$.

Theorem 5.4 (Riemann's inequality)**.** *Let $\mathcal{X}$ be a smooth projective and absolutely irreducible curve over $\mathbb{F}$. There exists a non-negative integer g such that for any divisor D on $\mathcal{X}$, we have*

$$\dim L_{\mathcal{X}}(D) \geqslant \deg D - g + 1.$$

The smallest integer g satisfying this property is called the genus *of $\mathcal{X}$, and is denoted by $g(\mathcal{X})$.*

Theorem 5.5 (Riemann-Roch Theorem)**.** *Let $\mathcal{X}$ be a curve of genus g and $D \in \mathrm{Div}(\mathcal{X})$. If $\deg D > 2g - 2$, then $\dim L_{\mathcal{X}}(D) = \deg D + 1 - g$.*

5.2.2 Definition of algebraic geometry codes

Definition 5.6 (Algebraic-geometry codes)**.** *Let $\mathcal{X}$ be an algebraic curve, $D \in \mathrm{Div}(\mathcal{X})$ and $\mathcal{P} \subset \mathcal{X}(\mathbb{F})$ of size $n \coloneqq |\mathcal{P}|$ such that $\mathrm{supp}(D) \cap \mathcal{P} = \varnothing$. We define the* Algebraic Geometry (AG) code *$C(\mathcal{X}, \mathcal{P}, D)$ as*

$$C(\mathcal{X}, \mathcal{P}, D) \coloneqq \left\{ f_{|\mathcal{P}} : \mathcal{P} \to \mathbb{F} \mid f \in L_{\mathcal{X}}(D) \right\}.$$

Assuming that $\deg D < n$ in Definition 5.6, the evaluation map $\mathrm{ev} : L_{\mathcal{X}}(D) \to \mathbb{F}^{\mathcal{P}}$ such that $C \coloneqq C(\mathcal{X}, \mathcal{P}, D)$ is the image of $L_{\mathcal{X}}(D)$ by ev is injective. In this case, the dimension of C is the dimension of $L_{\mathcal{X}}(D)$ and C has minimum distance at least $n - \deg D$. Moreover, if $\deg D \geqslant 2g(\mathcal{X}) - 1$, the dimension of C is given by the Riemann-Roch theorem (Theorem 5.5).

The divisor D will always be chosen so that the map ev is injective. To simplify notations, elements of C will be identified with functions in the Riemann-Roch space $L_{\mathcal{X}}(D)$.

We say that the C is a *one-point* AG code if the support of D consists in a single point.

Lemma 5.7 (AG codes over the projective line)**.** *Let $L = \{x_1, \ldots, x_n\} \subset \mathbb{F}$. Set $\mathcal{P} = \{(x_i \colon 1); x_i \in L\} \subset \mathbb{P}^1$ and $P_\infty = [1 \colon 0]$. Then the code $C(\mathbb{P}^1, \mathcal{P}, (k-1)P_\infty)$ is the Reed-Solomon code* $\mathsf{RS}[\mathbb{F}, L, k]$.

Proof. The rational functions with a pole of order less than k at infinity are polynomials of degree less than k. The Riemann-Roch space $L((k-1)P_\infty)$ is the space of rational functions with a pole of order $< k$ at infinity, which is exactly the space of polynomials of degree less than k. □

5.2.3 Additional material

We conclude this section with some additional preliminaries useful to present the results of this chapter.

Solvable groups. A finite group $\mathcal{G}$ is said to be *solvable* if there exists a sequence of subgroups of $\mathcal{G}$

$$\mathcal{G} = \mathcal{G}_0 \rhd \mathcal{G}_1 \rhd \cdots \rhd \mathcal{G}_r = 1,$$

such that $\mathcal{G}_{i+1}$ is a normal subgroup of $\mathcal{G}_i$ and each factor group $\mathcal{G}_i/\mathcal{G}_{i+1}$ is an abelian group for $i \in [\![0, r-1]\!]$. Such a sequence is called a *normal series*. If $\mathcal{G}$ is solvable, its cardinality equals the product of the sizes of the factor groups.

We recall that such a normal series is a *composition series* if $\mathcal{G}_i/\mathcal{G}_{i+1}$ is simple. If $\mathcal{G}$ is a finite group, we have that $\mathcal{G}$ is solvable if and only if $\mathcal{G}$ has a composition series whose factor groups are cyclic groups of prime order.

Automorphisms, group action and quotient curves. Let $\mathcal{X}$ be an algebraic curve. A group Γ is said to *act on the curve* $\mathcal{X}$ if Γ is a subgroup of the automorphism group $\mathrm{Aut}(\mathcal{X})$ of $\mathcal{X}$. The quotient curve $\mathcal{X}/\Gamma$ is the curve obtained by identifying points of $\mathcal{X}$ that lie in the same Γ-orbit.

The *stabilizer* of a point $P \in \mathcal{X}$ is the subgroup

$$\Gamma_P = \{\gamma \in \Gamma \mid \gamma \cdot P = P\} \subset \Gamma.$$

A divisor $D = \sum_P n_P P \in \mathrm{Div}(\mathcal{X})$ is said to be Γ-*invariant* is $n_P = n_{\gamma \cdot P}$ for all $P \in \mathcal{X}$ and $\gamma \in \Gamma$.

The action of Γ on $\mathcal{X}$ gives a projection $\pi : \mathcal{X} \to \mathcal{X}/\Gamma$ onto the quotient curve $\mathcal{X}/\Gamma$. A point $Q \in \mathcal{X}/\Gamma$ is called a *ramification point* if the number of preimages of Q by π is not equal to $|\Gamma|$. Equivalently, Q is a ramification point if one of its preimages has a non trivial stabilizer.

Hermitian tower. A tower of curves over $\mathbb{F}$ is an infinite sequence of curves $(\mathcal{X}_n)_{n \geqslant 0}$ and surjective maps $\pi_n : \mathcal{X}_{n+1} \twoheadrightarrow \mathcal{X}_n$ such that $g(\mathcal{X}_n) \to \infty$ as $n \to \infty$. Both the curves $\mathcal{X}_n$ and the maps π_n are defined over $\mathbb{F}$. For our applications, we will be particularly interested in recursive towers defined as follows. Let $f(X, Y) \in \mathbb{F}[X, Y]$ be an absolutely irreducible polynomial, namely irreducible over $\overline{\mathbb{F}}$. A tower $\mathcal{F}$ is recursively defined by $f(X, Y)$ if:

- The first curve $\mathcal{X}_0$ is the projective line $\mathbb{P}^1$ with affine coordinate X_1.
- The second curve $\mathcal{X}_1$ is the nonsingular projective model for the affine plane curve defined by $f(X_1, X_2) = 0$.
- The n-th curve $\mathcal{X}_n$ is the nonsingular projective model for the affine curve given by $f(X_1, X_2) = f(X_2, X_3) = \cdots = f(X_{n-1}, X_n) = 0$.

Specifically, we will consider towers of Hermitian curves.

Definition 5.8 (Hermitian tower)**.** *The* Hermitian tower *is the tower of curves $\mathcal{F}$ defined over $\mathbb{F} = \mathbb{F}_{q^2}$ by the recursive equation*

$$f(X, Y) = Y^q + Y - X^{q+1}.$$

The curve $\mathcal{X}_i$ of the Hermitian tower $\mathcal{F} = (\mathcal{X}_i)_{i \geqslant 0}$ over $\mathbb{F}_{q^2}$ has $q^{i+2} + 1$ $\mathbb{F}_{q^2}$-rational points.

Weierstrass gaps. Let $\mathcal{X}$ be a curve of genus $g > 1$ and P a $\mathbb{F}$-rational point on $\mathcal{X}$. The *Weierstrass semigroup* $H(P)$ of P is the set

$$H(P) \coloneqq \{n \in \mathbb{N} \mid \exists f \in \mathbb{F}(\mathcal{X}) \text{ with } \operatorname{div}_\infty(f) = nP\}.$$

A *(Weierstrass) gap* for P is an integer in $\mathbb{N} \backslash H(P)$, while any element of $H(P)$ is called a *nongap* for P.

We have that an integer $n \in \mathbb{N}$ is a nongap for P if and only if there exists a function with a pole of order n at P and no other poles. If n_1 and n_2 are nongaps for P, then $n_1 + n_2$ is also a nongap for P. We also have that n is a gap for P if and only if $\dim L(nP) = \dim L((n-1)P)$.

The number of gaps of a rational point is given by the genus g of the curve. Moreover, a gap n is always strictly smaller than $2g$. This property will be used to construct an IOPP for AG codes along the Hermitian tower in Section 5.6. More formally, we have the following theorem.

Theorem 5.9 (Weierstrass Gap Theorem)**.** *Suppose that $\mathcal{X}$ has genus $g > 0$ and $P \in \mathcal{X}$ is a rational point. Then $\mathbb{N} \backslash H(P) = \{i_1, \ldots, i_g\}$ with $1 = i_1 < i_2 < \cdots < i_g \leqslant 2g - 1$.*

Proof. [Sti09, Theorem 1.6.8]. □

5.3 Foldable AG codes

In this section, we display a workable setting for the construction of an IOPP system $(\mathscr{P}, \mathscr{V})$ to test whether a given function $f : \mathcal{P} \to \mathbb{F}$ is close to the evaluation of a function in a given Riemann-Roch space. As the idea is to iteratively reduce the problem of testing proximity to $C(\mathcal{X}, \mathcal{P}, D)$ to testing proximity to a smaller AG code, we introduce a sequence of suitable AG codes of decreasing length.

5.3.1 Sequence of curves

Fix a curve $\mathcal{X}$ defined over $\mathbb{F}$, a finite solvable group $\mathcal{G} \subseteq \operatorname{Aut}(\mathcal{X})$ and a normal series of $\bar{\mathcal{G}} \coloneqq (\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_r)$ of $\mathcal{G}$ such that

$$\mathcal{G} = \mathcal{G}_0 \rhd \mathcal{G}_1 \rhd \cdots \rhd \mathcal{G}_r = 1, \tag{5.2}$$

and $\mathcal{G}_i / \mathcal{G}_{i+1}$ is abelian.

For $i \in [\![0, r-1]\!]$, we denote by Γ_i the (abelian) factor group $\Gamma_i \coloneqq \mathcal{G}_i/\mathcal{G}_{i+1}$ and by p_i the order of Γ_i. We have that the cardinality of $\mathcal{G}$ equals

$$|\mathcal{G}| = \prod_{i=0}^{r-1} |\Gamma_i| .$$

The group Γ_0 acts on $\mathcal{X}_0 \coloneqq \mathcal{X}$, as factor group of $\mathcal{G}$. We thus define the quotient curve $\mathcal{X}_1 \coloneqq \mathcal{X}_0/\Gamma_0$. The group Γ_1 acts trivially on the orbits under Γ_0. Repeating the process for every $i \in [\![0, r-1]\!]$ defines a sequence of curves recursively as follows:

$$\mathcal{X}_0 := \mathcal{X} \text{ and } \mathcal{X}_{i+1} \coloneqq \mathcal{X}_i/\Gamma_i.$$

We set $F_i \coloneqq \mathbb{F}(\mathcal{X}_i)$ and we denote by $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ the canonical projection modulo the action of Γ_i. Even if the sequence of curves (5.3) depends on the derived series (5.2) of $\mathcal{G}$, the last curve $\mathcal{X}_r$ is always isomorphic to the quotient $\mathcal{X}/\mathcal{G}$:

$$\begin{array}{ccccccccccc} & {\scriptstyle \Gamma_0}\circlearrowright & & {\scriptstyle \Gamma_1}\circlearrowright & & & & {\scriptstyle \Gamma_i}\circlearrowright & & {\scriptstyle \Gamma_{i+1}}\circlearrowright & \\ \mathcal{X}_0 & \xrightarrow{\pi_0} & \mathcal{X}_1 & \xrightarrow{\pi_1} & \cdots & \xrightarrow{\pi_i} & \mathcal{X}_i & \xrightarrow{\pi_{i+1}} & \mathcal{X}_{i+1} & \longrightarrow \cdots \xrightarrow{\pi_{r-1}} & \mathcal{X}_r. \end{array} \tag{5.3}$$

Definition 5.10. *A sequence of curves constructed as above will be called a* $(\mathcal{X}, \bar{\mathcal{G}})$-sequence.

5.3.2 Definitions of foldable AG codes and balancing functions

Let $(\mathcal{X}_i)_{0\leqslant i\leqslant r}$ be a $(\mathcal{X}, \bar{\mathcal{G}})$-sequence. For any $i \in [\![0, r-1]\!]$, the factor group Γ_i which acts on the curve $\mathcal{X}_i$ is abelian of order p_i. For each $i \in [\![0, r]\!]$, we aim to define an AG code $C_i \subset \mathbb{F}^{\mathcal{P}_i}$ associated to a divisor $D_i \in \mathrm{Div}(\mathcal{X}_i)$ with an evaluation set $\mathcal{P}_i$. The rest of this subsection is dedicated to the choice of the sets $\mathcal{P}_i$ and the divisors D_i.

Evaluation points. From a set $\mathcal{P}_0 \subset \mathcal{X}(\mathbb{F})$, we want to recursively define a sequence of sets of points $(\mathcal{P}_i)_{1\leqslant i\leqslant r}$ so that $\mathcal{P}_i \subset \mathcal{X}_i(\mathbb{F})$ and $\mathcal{P}_{i+1} = \pi_i(\mathcal{P}_i)$. For our protocol, we need for each $i \in [\![0, r-1]\!]$ that every point in $\mathcal{P}_{i+1}$ admits exactly p_i preimages under π_i. Since the last curve $\mathcal{X}_r$ is isomorphic to the quotient $\mathcal{X}/\mathcal{G}$, it is necessary and sufficient that the first set $\mathcal{P}_0 \subset \mathcal{X}_0$ is a union of $\mathcal{G}$-orbits of size $|\mathcal{G}|$, i.e. that $\mathcal{G}$ acts freely on $\mathcal{P}_0$.

Divisors. Fix a divisor $D_0 \in \mathrm{Div}(\mathcal{X}_0)$ that is globally Γ_0-invariant. This way, the support of D_0 does not meet the set $\mathcal{P}_0$. For the sake of simplicity, we will assume that D_0 is in fact supported by Γ_0-fixed points.

Notation 2. *For a divisor* $D = \sum n_P P \in \mathrm{Div}(\mathcal{X})$ *and a positive integer* n*, we denote by* $\left\lfloor \frac{1}{n} D \right\rfloor \in \mathrm{Div}(\mathcal{X})$ *the divisor defined by*

$$\left\lfloor \frac{1}{n} D \right\rfloor := \sum \left\lfloor \frac{n_P}{n} \right\rfloor P.$$

Definition 5.11. *Let* $i \in [\![0, r-1]\!]$*. Fix a divisor* $D_i \in \mathrm{Div}(\mathcal{X}_i)$ *and a function* $\mu_i \in F_i$ *. We say that* μ_i partitions $L_{\mathcal{X}_i}(D_i)$ *(with respect to the action of* Γ_i*) if*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{p_i-1} \mu_i^j \pi_i^* L_{\mathcal{X}_{i+1}}(E_{i,j}) \tag{5.4}$$

with

$$E_{i,j} \coloneqq \left\lfloor \frac{1}{p_i} \pi_{i*}(D_i + j \operatorname{div}_{\mathcal{X}_i}(\mu_i)) \right\rfloor \in \operatorname{Div}(\mathcal{X}_{i+1}) \text{ for } j \in [\![0, p_i - 1]\!]. \tag{5.5}$$

For our applications, we wish to be able to define a sequence of divisors (D_i) that have the following properties:

- the divisor D_i is supported by Γ_i-invariant points;
- for each divisor D_i, its associated Riemann-Roch admits a nice decomposition like (5.4);
- at each step, a divisor D_{i+1} needs to be "compatible" with D_i and the decomposition of $L_{\mathcal{X}_i}(D_i)$ in the sense of Definition 5.12 below.

Definition 5.12 (Compatible divisors and balancing functions)**.** *Let $i \in [\![0, r-1]\!]$. Fix a divisor $D_i \in \operatorname{Div}(\mathcal{X}_i)$ and a function $\mu_i \in F_i$ such that μ_i partitions $L_{\mathcal{X}_i}(D_i)$. A divisor $D_{i+1} \in \operatorname{Div}(\mathcal{X}_{i+1})$ is said to be* compatible with (D_i, μ_i) *if both assertions hold.*

1. *for every $j \in [\![0, p_i - 1]\!]$, $E_{i,j} \leqslant D_{i+1}$,*
2. *for every $j \in [\![0, p_i - 1]\!]$, there exists a function $\nu_{i+i,j} \in \mathbb{F}(\mathcal{X}_{i+1})$ such that*

$$\operatorname{div}_\infty\left(\nu_{i+i,j}\right) = D_{i+1} - E_{i,j}. \tag{5.6}$$

The functions $\nu_{i+i,j}$ are called balancing functions*.*

In Definition 5.12, the first item implies that $L(E_{i,j}) \subseteq L(D_{i+1})$. The second one means that for every $f_j \in L(E_{i,j})$, the function $\nu_{i+1,j} f_j$ lies in $L(D_{i+1})$.

We have now described all the key components to formally define the notion of foldable AG codes.

Definition 5.13 (Foldable AG codes)**.** *Let $C = C(\mathcal{X}, \mathcal{P}, D)$ be an AG-code. This code is said to be* foldable *if the following conditions are satisfied.*

1. *There exists a finite solvable group $\mathcal{G} \in \operatorname{Aut}(\mathcal{X})$ that acts freely on $\mathcal{P}$: a composition series of $\mathcal{G}$* (5.2) *provides a $(\mathcal{X}, \bar{\mathcal{G}})$-sequence of curves $(\mathcal{X}_i)$;*
2. *There exists $e \in (0,1)$ such that $|\mathcal{G}| > |\mathcal{P}|^e$;*
3. *There exist some sequences $(\mu_i) \in F_i$ and $(D_i) \in \operatorname{Div}(\mathcal{X}_i)$ such that $D_0 = D$ and for every $i \in [\![0, r-1]\!]$, all the following properties hold:*
   (a) *the divisors D_i are supported by Γ_i-fixed points,*
   (b) *the function μ_i partitions $L_{\mathcal{X}_i}(D_i)$ (Definition 5.11),*
   (c) *D_{i+1} is (D_i, μ_i)-compatible (Definition 5.12).*

Remark 5.14. *The second requirement given in Definition 5.12 is definitely compelling and requires some geometric knowledge about the curves $\mathcal{X}_i$. Indeed, on a general curve, not every effective divisor is the poles locus of a function, and characterizing which effective divisors arise this way is at the heart of the Weierstrass gaps theory. Nonetheless, the existence of the balancing functions $\nu_{i+1,j}$ happens to be the main ingredient in Lemma 5.24, which takes a prominent role in the design of our IOPP.*

To prevent the relative minimum distance of the code C_{i+1} from collapsing and thence ensure a good soundness of the protocol designed in Section 5.4, one may be tempted to take D_{i+1} as one of the divisors $E_{i,j}$ (5.5) *that appear in the decomposition* (5.4) *of $L_{\mathcal{X}_i}(D)$. However the Weierstrass gaps theory indicates that balancing functions exist only when choosing a (D_i, μ_i)-compatible divisor D_{i+1} whose degree may be unexpectedly substantial. Therefore, to broaden the spectrum of foldable codes, we do not make this additional hypothesis.*

5.3.3 Reed-Solomon codes as foldable AG codes

Recall that RS codes are AG codes on the projective line (Lemma 5.7). Assume that $\mathbb{F}$ has odd characteristic. In this case, the decomposition (5.4) is nothing but the splitting of a polynomial into an even part and an odd part, which plays a crucial role in the FRI protocol.

To make both points of views coincide, let us consider the involution $\gamma : [X_0 : X_1] \mapsto [-X_0 : X_1]$. It generates a group isomorphic to $\mathbb{Z}/2\mathbb{Z}$ and the quotient of $\mathbb{P}^1$ by this group is obtained as the image by $\pi : [X_0 : X_1] \mapsto [X_0^2 : X_1^2]$.

The divisor $D \coloneqq dP_\infty$ is invariant under γ. Let us choose μ as the function $x = \frac{X_0}{X_1}$. We have $\mathrm{div}(x) = P_0 - P_\infty$ with $P_0 = [0:1]$ and $P_\infty = [1:0]$. Noticing that $\pi_*(P_\infty) = P_\infty$ and $\pi_*(P_0) = P_0$, we get

$$\left\lfloor \frac{1}{2}\pi_*(D + (x)) \right\rfloor = \left\lfloor \frac{1}{2}((d-1)P_\infty + P_0) \right\rfloor = \left\lfloor \frac{d-1}{2} \right\rfloor P_\infty,$$

and the Riemman-Roch space $L_{\mathbb{P}^1}(dP_\infty)$ is split into two parts:

$$L_{\mathbb{P}^1}(dP_\infty) = \pi^* L_{\mathbb{P}^1}\left(\left\lfloor \frac{d}{2} \right\rfloor P_\infty\right) + x\pi^* L_{\mathbb{P}^1}\left(\left\lfloor \frac{d-1}{2} \right\rfloor P_\infty\right).$$

We recover the decomposition of a polynomial of degree d into even and odd parts of respective degrees $\left\lfloor \frac{d}{2} \right\rfloor$ and $\left\lfloor \frac{d-1}{2} \right\rfloor$.

Remark 5.15. *The function μ is not unique: any odd polynomial of x would make a suitable choice for μ.*

Now, let us remark that the RS code

$$V \coloneqq \left\{ f \in \mathbb{F}^{\mathcal{P}}; \deg f \leqslant d \right\} = C(\mathbb{P}^1, \mathcal{P}, dP_\infty)$$

is a foldable AG code, for any $\mathcal{P} \subset \mathbb{F}$ of size $|\mathcal{P}| = 2^r$ for a certain integer r and any degree bound d. We shall then retrieve the construction of the RS proximity test of [BBHR18a].

Firstly, the finite solvable $\mathbb{Z}/2^r\mathbb{Z}$ of size $|\mathcal{P}|$ acts on $\mathbb{P}^1$ via $[X_0 : X_1] \mapsto [X_0, \xi X_1]$, where ξ is a primitive 2^r-th root unity. It clearly fulfils the two first items of Definition 5.13. When considering its composition series

$$\mathbb{Z}/2^r\mathbb{Z} \rhd \mathbb{Z}/2^{r-1}\mathbb{Z} \rhd \cdots \rhd 1 \tag{5.7}$$

and the action of the corresponding factor group $\Gamma = \langle \gamma \rangle \simeq \mathbb{Z}/2\mathbb{Z}$, we obtain a trivial sequence of curves $(\mathcal{X}_i)$ where $\mathcal{X}_i = \mathbb{P}^1$ for all $i \in [\![0, r]\!]$. Next, consider the sequence (μ_i) with $\mu_i = \mu = x \coloneqq \frac{X_1}{X_0}$, then $\gamma\mu = -\mu$. Set $d_0 \coloneqq d$, and for any $i \in [\![0, r-1]\!]$, $d_{i+1} \coloneqq \left\lfloor \frac{d_i}{2} \right\rfloor$. Note that there exists $r' < r$ such that $d_{r'}, \ldots, d_r$ are all equal to 0. Setting $D_i = \left\lfloor \frac{d_i}{2} \right\rfloor P_\infty$, we have Γ_i-invariant divisors fulfilling the compatibility condition given in Definition 5.12, by letting $\nu_{i+1,j}$ to be the constant function equal to 1 if $\left\lfloor \frac{d_i}{2} \right\rfloor = \left\lfloor \frac{d_i - 1}{2} \right\rfloor$, and $\nu_{i+1,j} = x$ otherwise.

5.3.4 Kani's theorem

The first requirement to make a sequence of codes foldable is the splitting of the Riemann-Roch spaces as in (5.4), which mimics the decomposition in odd and even parts of univariate polynomials. Under some additional hypotheses, a decomposition like (5.4) always exists. Let us detail this framework.

Let $\mathcal{X}$ be a smooth irreducible curve over a field $\mathbb{F}$ and let Γ be a cyclic group of order m generated by an element γ. Assume that m and the characteristic of $\mathbb{F}$ are coprime and consider $\zeta \in \bar{\mathbb{F}}$ a primitive m^{th} root of unity, lying in some algebraic closure $\bar{\mathbb{F}}$ of $\mathbb{F}$.

Set $\mathcal{Y} := \mathcal{X}/\Gamma$ and $\pi : \mathcal{X} \to \mathcal{Y}$ be the canonical projection morphism.

Fix a Γ-invariant divisor $D \in \mathrm{Div}(\mathcal{X})$. We want to exhibit a relation between the Riemann-Roch space $L_{\mathcal{X}}(D)$ and some Riemann-Roch spaces on $\mathcal{Y}$. The group Γ acts on the vector space $L_{\mathcal{X}}(D)$ via $\gamma \cdot f = f \circ \gamma$. By representation theory,

$$L_{\mathcal{X}}(D) = \bigoplus_{j=0}^{m-1} L_{\mathcal{X}}(D)_j,$$

where $L_{\mathcal{X}}(D)_j := \{g \in L_{\mathcal{X}}(D) \mid \gamma \cdot g = \zeta^j g\}$.

One of the key ingredients of this section is a theorem due to Kani [Kan86], which we reformulate here in the case where Γ is cyclic.

Theorem 5.16 ([Kan86])**.** *Assume that $\Gamma = \langle\gamma\rangle$ is cyclic of order m, coprime with $|\mathbb{F}|$. The two following statements hold.*

1. *There exists a function $\mu \in \bar{\mathbb{F}}(\mathcal{X})$ such that $\gamma \cdot \mu = \zeta\mu$.*
2. *For any Γ-invariant divisor $D \in \mathrm{Div}(\mathcal{X})$, when considering the Riemann-Roch spaces over the algebraic closure $\bar{\mathbb{F}}$, we have*

$$L_{\mathcal{X}}(D)_j \otimes \bar{\mathbb{F}} \simeq \mu^j \pi^* \left(L_{\mathcal{Y}} \left(\left\lfloor \frac{1}{m} \pi_* \left(D + j \,\mathrm{div}(\mu) \right) \right\rfloor \right) \otimes \bar{\mathbb{F}} \right). \tag{5.8}$$

Remark 5.17. *A function μ_i provided by Theorem 5.16 satisfies Definition 5.11. We will see in Section 5.6 that such a decomposition may exist without the hypotheses of Theorem 5.16, for example if the characteristic of $\mathbb{F}$ divides $|\mathcal{G}|$.*

Remark 5.18. *If the function μ is defined over the base field $\mathbb{F}$ then the decomposition* (5.8) *is valid when considering $\mathbb{F}$-vector spaces:*

$$L_{\mathcal{X}}(D)_j \simeq \mu^j \pi^* \left(L_{\mathcal{Y}} \left(\left\lfloor \frac{1}{m} \pi_* \left(D + j \,\mathrm{div}(\mu) \right) \right\rfloor \right) \right).$$

In practical instantiations, we are always able to choose μ defined over $\mathbb{F}$, even when ζ does not belong to $\mathbb{F}$. Thus, such a decomposition holds even if there is no m^{th} primitive root in the base field $\mathbb{F}$. However, as the evaluation set $\mathcal{P}$ is formed of orbits of size $|\mathcal{G}|$, our applications will require this primitive root to belong to $\mathbb{F}$, as it is the case for Kummer curves (see Section 5.5).

To handle the divisors appearing in the decomposition above, we need to get a better grasp on the zeroes and the poles of the function μ, including the ramification points of π according to the following lemma.

Lemma 5.19. *Assume that $\Gamma = \langle\gamma\rangle$ is a cyclic group of order m. Let P be a point of $\mathcal{X}$ whose stabilizer Γ_P is non trivial. Then $P \in \mathrm{supp}(\mu)$.*

Proof. By hypothesis, there exists $j \in [\![1, m-1]\!]$ such that $\gamma^j \in \Gamma_P$. Then

$$\begin{aligned} (\gamma^j \cdot \mu)(P) &= \zeta^j \mu(P) && \text{by definition of } \mu \text{ in Th. 5.16,} \\ &= \mu(P) && \text{because } \gamma^j \in \Gamma_P. \end{aligned}$$

Since $\zeta^j \neq 1$, the point P is either a pole or a zero of μ. □

5.4 IOPP for foldable AG codes

Now that we have determined the needed properties of an AG code to be foldable, we construct folding operators (Definition 3.1 and Definition 3.4) for them. This yields an IOPP for foldable AG codes, using Construction 3.5.

5.4.1 Definition of folding operators and properties

Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a code satisfying Definition 5.13. We consider its associated $(\mathcal{X}, \mathcal{G})$-sequence of curves $(\mathcal{X}_i)$ and its sequence of divisors (D_i).

To test proximity of a function $f^{(0)} : \mathcal{P}_0 \to \mathbb{F}$ to C_0, we aim to inductively reduce the problem to a smaller one, consisting of testing proximity to the code $C_i = C(\mathcal{X}_i, \mathcal{P}_i, D_i)$. Broadly speaking, our goal is to define from any function $f^{(i)} : \mathcal{P}_i \to \mathbb{F}$ a function $f^{(i+1)} : \mathcal{P}_{i+1} \to \mathbb{F}$ such that the relative distance $\Delta(f^{(i+1)}, C_{i+1})$ is roughly equal to $\Delta(f^{(i)}, C_i)$.

Fix $i \in [\![0, r-1]\!]$ and let $f : \mathcal{P}_i \to \mathbb{F}$ be an arbitrary function.

Notation 3 (Interpolation polynomial)**.** *For each $P \in \mathcal{P}_{i+1}$, let us denote $S_P \coloneqq \pi_i^{-1}(\{P\})$ the set of p_i distinct preimages of P and consider*

$$I_{f,P}(X) \coloneqq \sum_{j=0}^{p_i-1} X^j a_{j,P} \tag{5.9}$$

the univariate polynomial over $\mathbb{F}$ of degree less than p_i which interpolates the set of points

$$\left\{(\mu_i(\hat{P}), f(\hat{P})); \hat{P} \in S_P\right\}.$$

Then for every $j \in [\![0, p_i - 1]\!]$, we define the function

$$f_j : \begin{cases} \mathcal{P}_{i+1} & \to & \mathbb{F}, \\ P & \mapsto & a_{j,P}. \end{cases} \tag{5.10}$$

Given $f : \mathcal{P}_i \to \mathbb{F}$, the idea is to define p_i functions $f_j : \mathcal{P}_{i+1} \to \mathbb{F}$, where $|\mathcal{P}_{i+1}| = \frac{|\mathcal{P}_i|}{p_i}$ such that f corresponds to the evaluation of a function in $L(D_i)$ if and only if each f_j coincides with a function in $L(E_{i,j}) \subset L(D_{i+1})$. Instead of testing for each $j \in [\![0, p_i - 1]\!]$ whether $f_j \in C_{i+1}$, we reduce those p_i claims to a single one, by taking a random linear combination of the f_j's, which we referred to as a *folding of f*. By linearity of the codes, such a combination of the f_j's belongs to C_{i+1} whenever $f \in C_i$ (see Proposition 5.22 below). However, for soundness analysis, one needs to ensure that no f_j corresponds to a function lying in $L(D_{i+1}) \backslash L(E_{i,j})$. Some safeguards are embedded into the folding operation by introducing the balancing functions $\nu_{i+1,j}$ from Definition 5.12 in the second term of the sum in (5.11).

Definition 5.20 (Folding operator)**.** *For any $\boldsymbol{z} = (z_1, z_2) \in \mathbb{F}^2$, we define the* folding of f *to be the function* **Fold** $[f, \boldsymbol{z}] : \mathcal{P}_{i+1} \to \mathbb{F}$ *such that*

$$\textbf{Fold}\,[f, \boldsymbol{z}] \coloneqq \sum_{j=0}^{p_i-1} z_1^j f_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} f_j \tag{5.11}$$

where the functions f_j are defined in Equation (5.10) *and the functions $\nu_{i+j,j}$ in Definition 5.12.*

Our aim is to prove that the folding operators satisfy three key properties: local computability, completeness, and distance preservation. This will enable us to invoke Theorem 3.8 for the completeness and soundness of our AG-IOPP.

Given the p_i points $((\mu_i(\hat{P}), f(\hat{P})))_{\hat{P}\in S_P}$, one can determine the coefficients $(a_{j,P})_{0\leqslant j<p}$ of $I_{f,P}$ defined in (5.9) by polynomial interpolation. Recalling that for each $P \in \mathcal{P}_{i+1}$, we have $f_j(P) = a_{j,P}$, we get the following lemma. This lemma will allow to obtain fast prover time and verifier decision complexity.

Lemma 5.21 (Locality)**.** *Let $z \in \mathbb{F}^2$. For each $P \in \mathcal{P}_{i+1}$, the value of* **Fold** $[f, z]\,(P)$ *can be computed with exactly p_i queries to f, namely at the points $\pi_i^{-1}(\{P\})$.*

Proposition 5.22 (Completeness)**.** *Let $z \in \mathbb{F}^2$. If $f \in C_i$, then* **Fold** $[f, z] \in C_{i+1}$.

Proof. Write $z = (z_1, z_2)$. If $f \in C_i$, it coincides with a function of $L(D_i)$. By definition of the divisors $E_{i,j}$ and Theorem 5.16, there exist some functions $\widetilde{f}_j \in L(E_{i,j})$ such that

$$f = \sum_{j=0}^{p_i-1} \mu_i^j \widetilde{f}_j \circ \pi_i.$$

Let $P \in \mathcal{P}_{i+1}$, for any $\hat{P} \in S_P$,

$$\textbf{Fold}\left[f, (\mu_i(\hat{P}), 0)\right](P) = I_{f,P}(\mu_i(\hat{P})) = f(\hat{P}) = \sum_{j=0}^{p_i-1} \mu_i(\hat{P})^j \widetilde{f}_j(P).$$

Moreover, for any $P \in \mathcal{P}_{i+1}$, polynomials $I_{f,P}(X)$ and **Fold** $[f, (X, 0)]\,(P) \in \mathbb{F}[X]$ are of degree less than p_i and agree on $\left\{\mu_i(\hat{P}); \hat{P} \in S_P\right\}$ of size p_i, therefore they are equal. In particular,

$$\textbf{Fold}\left[f, (\mu_i(\hat{P}), 0)\right](P) = \sum_{j=0}^{p_i-1} \mu_i(\hat{P})^j f_j(P).$$

Thus, for all $P \in \mathcal{P}_{i+1}$,

$$\sum_{j=0}^{p_i-1} \mu_i(\hat{P})^j (\widetilde{f}_j(P) - f_j(P)) = 0$$

and the polynomial

$$\sum_{j=0}^{p_i-1} X^j (\widetilde{f}_j(P) - f_j(P))$$

of degree less than p_i is zero on at least $\left|\left\{\mu_i(\hat{P}); P \in \mathcal{P}_{i+1}\right\}\right| = p_i$ points. Hence, for every $j \in [\![0, p_i - 1]\!]$, the function f_j defined in Equation (5.10) coincides with $\widetilde{f}_j$ and

$$\textbf{Fold}\,[f, z] \coloneqq \sum_{j=0}^{p_i-1} z_1^j \widetilde{f}_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} \widetilde{f}_j$$

where $\widetilde{f}_j \in L(E_{i,j}) \subseteq L(D_{i+1})$ and $\nu_{i+1,j} f_j \in L(D_{i+1})$, by definition of the divisors $E_{i,j}$, D_{i+1} and the functions $\nu_{i+1,j}$ (see Definition 5.12). Thus each term of **Fold** $[f, z]$ lies in the vector space C_{i+1}, which concludes the proof. □

We now discuss the effect of the folding operation on a function which is far from the code. Roughly speaking, we want to show that, if f is δ-far from C_i, then the folding $\mathbf{Fold}[f,z]$ of f is almost δ-far from C_{i+1} with high probability over $z \in \mathbb{F}^2$.

Proposition 5.23. *Fix $i \in [\![0, r-1]\!]$. For any weight function $\varphi_i : \mathcal{P}_i \to [0,1]$, define $\varphi_{i+1} : \mathcal{P}_{i+1} \to [0,1]$ by*

$$\forall P \in \mathcal{P}_{i+1},\ \varphi_{i+1}(P) \coloneqq \frac{1}{p_i} \sum_{\hat{P} \in S_P} \varphi_i(\hat{P}).$$

Let λ_i be the minimal relative distance of C_i. Fix $\varepsilon \in (0,1[$ and $\delta < \min\left(J_\varepsilon^{p_i}(\lambda_i), \frac{1}{2}\left(\lambda_i + \frac{\varepsilon}{2}\right)\right)$. For any function $f : \mathcal{P}_i \to \mathbb{F}$ such that $\mathsf{agree}_{\varphi_i}(f, C_i) < 1 - \delta$, we have

$$\Pr_{z \in \mathbb{F}^2}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}[f,z], C_{i+1}) > 1 - \delta + \varepsilon\right] \leqslant \frac{1}{|\mathbb{F}|}\left(p_i + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_i}.$$

Proving Proposition 5.23 requires the lemma stated next. We prove Proposition 5.23, then prove Lemma 5.24.

Lemma 5.24. *Let $i \in [\![0, r-1]\!]$, $D_i \in \mathrm{Div}(\mathcal{X}_i)$ and $\mu_i \in \mathbb{F}(\mathcal{X}_i)$ satisfying* (5.11). *Consider a divisor $D_{i+1} \in \mathrm{Div}(\mathcal{X}_{i+1})$ that is (D_i, μ_i)-compatible in the sense of Definition 5.12.*

Fix $j \in [\![0, p_i - 1]\!]$. Then a function $g \in \mathbb{F}(\mathcal{X}_{i+1})$ belongs to $L(E_{i,j})$ if and only if both functions g and $g\nu_{i+1,j}$ belong to $L(D_{i+1})$.

Proof of Proposition 5.23. Let $f : \mathcal{P}_i \to \mathbb{F}$ be an arbitrary function. According to Equation (5.10), there exist p_i function $f_j : \mathcal{P}_{i+1} \to \mathbb{F}$ such that for any $z = (z_1, z_2) \in \mathbb{F}^2$,

$$\mathbf{Fold}[f,z] = \sum_{j=0}^{p_i-1} z_1^j f_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} f_j.$$

Rewrite $\mathbf{Fold}[f,z]$ as a polynomial function in z_2, i.e. $\mathbf{Fold}[f,z] = f_{z_1} + z_2 f'_0 + z_2^2 f'_1 + \cdots + z_2^{p_i} f'_{p_i-1}$ where we set $f_{z_1} \coloneqq \sum_{j=0}^{p_i-1} z_1^j f_j$ and $f'_j \coloneqq \nu_{i+1,j} f_j$. Finally, set

$$K_0 \coloneqq \frac{p_i - 1}{|\mathbb{F}|}\left(\frac{4}{\varepsilon}\right)^{p_i} \qquad \text{and} \qquad K_1 \coloneqq \frac{p_i}{|\mathbb{F}|}\left(\frac{4}{\varepsilon}\right)^{p_i+1}.$$

Let us prove the corollary by contrapositive: assume that

$$\Pr_{z \in \mathbb{F}^2}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}[f,z], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0 + K_1.$$

Thus we have

$$\Pr_{z_1 \in \mathbb{F}}\left[\Pr_{z_2 \in \mathbb{F}}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}[f,z], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0\right] > K_1.$$

Fix $z_1 \in \mathbb{F}$ such that

$$\Pr_{z_2 \in \mathbb{F}}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}[f,z], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0.$$

By Proposition 3.16, there exist $v_{z_1}, v'_1, \ldots, v'_{p_i-1} \in C_{i+1}$ and $\mathcal{T}' \subset \mathcal{P}$ such that

- $\sum_{P\in\mathcal{T}'}\theta(P)\geqslant(1-\delta+\frac{\varepsilon}{2})\,|\mathcal{P}_{i+1}|$,
- $v_{z_1|\mathcal{T}'}=f_{z_1|\mathcal{T}'}$,
- for each $j\in[\![1,p_i-1]\!]$, $v'_{j|\mathcal{T}'}=f'_{j|\mathcal{T}'}$.

In particular,

$$\mathsf{agree}_{\varphi_{i+1}}(f_{z_1},C_{i+1})\geqslant\mathsf{agree}_{\varphi_{i+1}}(f_{z_1},v_{z_1})=\frac{1}{|\mathcal{P}_{i+1}|}\sum_{P\in\mathcal{T}'}\theta(P)\geqslant 1-\delta+\frac{\varepsilon}{2}.$$

It means that

$$\begin{aligned}\Pr_{z_1\in\mathbb{F}}\left[\mathsf{agree}_{\varphi_{i+1}}(f_{z_1},C_{i+1})\geqslant 1-\delta+\frac{\varepsilon}{2}\right]&\geqslant\Pr_{z_1\in\mathbb{F}}\left[\Pr_{z_2\in\mathbb{F}}\left[\mathsf{agree}_{\varphi_{i+1}}(\mathbf{Fold}\left[f,\boldsymbol{z}\right],C_{i+1})>1-\delta+\varepsilon\right]>K_0\right]\\&>K_1.\end{aligned}$$

The polynomial form of f_{z_1} in z_1 enables us to reapply Proposition 3.16: there exist $v_0,v_1,\ldots,v_{p_i-1}\in C_{i+1}$ and $\mathcal{T}\subset\mathcal{P}$ such that

- $\sum_{P\in\mathcal{T}}\theta(P)\geqslant(1-\delta)\,|\mathcal{P}_{i+1}|$,
- for each $j\in[\![0,p_i-1]\!]$, $v_{j|\mathcal{T}}=f_{j|\mathcal{T}}$.

On $\mathcal{T}'\cap\mathcal{T}$, we thus have

$$v'_{j|\mathcal{T}'\cap\mathcal{T}}=f'_{j|\mathcal{T}'\cap\mathcal{T}}=(\nu_{i+1,j}f_j)_{|\mathcal{T}'\cap\mathcal{T}}=(\nu_{i+1,j}v_j)_{|\mathcal{T}'\cap\mathcal{T}}.$$

The cardinality of $\mathcal{T}'\cap\mathcal{T}$ satisfies

$$|\mathcal{T}'\cap\mathcal{T}|=|\mathcal{T}'|+|\mathcal{T}|-|\mathcal{T}'\cup\mathcal{T}|\geqslant\sum_{P\in\mathcal{T}'}\theta(P)+\sum_{P\in\mathcal{T}}\theta(P)-|\mathcal{P}_{i+1}|\geqslant(1-2\delta+\frac{\varepsilon}{2})|\mathcal{P}_{i+1}|.$$

The assumption on δ ensures that $2\delta-\frac{\varepsilon}{2}<\lambda_{i+1}$ where λ_{i+1} is the minimal distance of C_{i+1}. Hence, for every $j\in[\![0,p_i-1]\!]$, the evaluations of v'_j and $\nu_{i+1,j}v_j$ on $\mathcal{P}_{i+1}$ are equal. They are codewords of C_{i+1}, thus this implies that both functions v_j and $\nu_{i+1,j}v_j$ belong to $L(D_{i+1})$. By Lemma 5.24, we get that the function v_j lies in $L(E_{i,j})$.

Now let us define $v:\mathcal{P}_i\to\mathbb{F}$ by

$$\forall Q\in\mathcal{P}_i,\ v(Q)\coloneqq\sum_{j=0}^{p_i-1}\mu_i^j(Q)v_j\circ\pi_i(Q).$$

By definition of the divisors $E_{i,j}$ (5.5), the function v belong to $L(D_i)$. Now let us prove that it agrees with f on $S_{\mathcal{T}}\coloneqq\bigsqcup_{P\in\mathcal{T}}S_P$. For any $P\in\mathcal{T}$ and $\hat{P}\in S_P$, we have

$$\begin{aligned}f(\hat{P})=I_{f,P}(\mu_i(\hat{P}))&=\sum_{j=0}^{p_i-1}\mu_i(\hat{P})^jf_j(P)&&\text{by definition of } I_{f,P},\\&=\sum_{j=0}^{p_i-1}\mu_i(\hat{P})^jv_j\circ\pi_i(\hat{P})&&\text{since } f_{j|\mathcal{T}}=v_{j|\mathcal{T}} \text{ and } P=\pi_i(\hat{P}),\\&=v(\hat{P}).\end{aligned}$$

As a result, since $v \in C_i$, we can conclude that

$$\mathsf{agree}_{\varphi_i}(f, C_i) \geqslant \mathsf{agree}_{\varphi_i}(f, v) \geqslant \frac{1}{|\mathcal{P}_i|} \sum_{P \in \mathcal{T}} \sum_{\hat{P} \in S_P} \eta(\hat{P}) = \frac{1}{|\mathcal{P}_{i+1}|} \sum_{P \in \mathcal{T}} \theta(P) \geqslant 1 - \delta.$$

□

Proof of Lemma 5.24. Assume that $g \in L(E_{i,j})$. Then the second and third items of Definition 5.12 ensure that g and $g\nu_{i+1,j}$ lie in $L(D_{i+1})$.

Conversely, assume that g and $g\nu_{i+1,j}$ belong to $L(D_{i+1})$ and write $D_{i+1} = \sum n_P P$. The hypotheses on g imply that $g \in L(D_{i+1}) \cap L(D_{i+1} - (\nu_{i+1,j}))$. By [MP93, Lemma 2.6], the function g belongs to $L(D'_{i+1})$, where the divisor D'_{i+1} is defined by

$$D'_{i+1} \coloneqq \sum_P n'_P P \qquad \text{where } n'_P \coloneqq \min(n_P, n_P + v_P(\nu_{i+1,j})).$$

Then $D'_{i+1} = D_{i+1} - \mathrm{div}_\infty\left(\nu_{i+1,j}\right) = E_{i,j}$ by the third item of Definition 5.12. □

5.4.2 Foldable AG codes admit efficient IOPP

Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code over an alphabet $\mathbb{F}$. Given a family of folding operators defined as per Definition 5.20, Construction 3.5 yields an IOPP for C_0, which is abstracted from the FRI protocol of [BBHR18a]. We informally describe the IOPP system $(\mathscr{P}, \mathscr{V})$ for testing proximity of a function $f^{(0)} : \mathcal{P}_0 \to \mathbb{F}$ to C_0, then give its properties. A formal description will be provided in Section 5.7 for instantiations with concrete AG codes.

As in the FRI protocol, the IOPP is divided in two phases, referred to as COMMIT and QUERY. Before any interaction, $\mathscr{P}$ and $\mathscr{V}$ agree on:

- a $(\mathcal{X}, \bar{\mathcal{G}})$-sequence of curves $(\mathcal{X}_i)_{0 \leqslant i \leqslant r}$ of length r.
- a sequence of codes $(C_i)_{0 \leqslant i \leqslant r}$ where for each $i \in \{0, \dots, r\}$, $C_i = (\mathcal{X}_i, \mathcal{P}_i, D_i)$ and $\mathcal{X}_i, \mathcal{P}_i$ and D_i are defined as per Section 5.3,
- a sequence of functions $(\mu_i)_{0 \leqslant i < r} \in \mathbb{F}(\mathcal{X}_i)$ satisfying Definition 5.11,
- a sequence of balancing functions $(\nu_{i+1})_{0 \leqslant i < r}$ of p_i-tuples of functions in $\mathbb{F}(\mathcal{X}_{i+1})$ such that $\nu_{i+1} = (\nu_{i+1,j})_{0 < j < p_i}$ and $\nu_{i+1,j}$ satisfies (5.6).

We recall that the choice of a sequence $(\mathcal{X}_i)_{0 \leqslant i \leqslant r}$ induces a sequence of projections $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$.

- The COMMIT phase is an interaction over r rounds between $\mathscr{P}$ and $\mathscr{V}$. For each round $i \in [\![0, r-1]\!]$, the verifier samples a random challenge $z^{(i)} \in \mathbb{F}^2$. As an answer, the prover gives oracle access to function $f^{(i+1)} : \mathcal{P}_{i+1} \to \mathbb{F}$, which is expected to be equal to $\textbf{Fold}\left[f^{(i)}, z^{(i)}\right]$. To compute the values of $f^{(i+1)}$ on $\mathcal{P}_{i+1}$, an honest prover $\mathscr{P}$ exploits the fact that the folding of $f^{(i)}$ is locally computable (Lemma 5.21). Namely, for each $P \in \mathcal{P}_{i+1}$, $\mathscr{P}$ computes the coefficients $(a_{j,P})_{0 \leqslant j < p}$ of $I_{f^{(i)},P} \in \mathbb{F}[X]$ from ${f^{(i)}}_{|S_P}$, evaluates $\nu_{i+1,j}$ at P, and set

$$\textbf{Fold}\left[f^{(i)}, z^{(i)}\right](P) \coloneqq \sum_{j=0}^{p_i - 1} \left(z_1^{(i)}\right)^j a_{j,P} + \sum_{j=0}^{p_i - 1} \left(z_2^{(i)}\right)^{j+1} \nu_{i+1,j}(P) a_{j,P}.$$

– During the QUERY phase, one of the two tasks of the verifier $\mathscr{V}$ is to check that each pair of successive oracle functions $(f^{(i)}, f^{(i+1)})$ is consistent. A standard idea is to check that the equality

$$f^{(i+1)} = \textbf{Fold}\left[f^{(i)}, z^{(i)}\right] \tag{5.12}$$

holds at a random point in $\mathcal{P}_{i+1}$. By leveraging the local property of the folding operator, such a test requires only p_i queries to $f^{(i)}$ and 1 query to $f^{(i+1)}$. As in [BBHR18a], we call this step of verification a *round consistency test*. The verifier begins by sampling at random $Q_0 \in \mathcal{P}_0$ and once this is done, all the locations of the round consistency tests run inside the current **query test** are determined. More specifically, for each round i, $\mathscr{V}$ defines $Q_{i+1} \coloneqq \pi_i(Q_i)$ to be the random point where Equation (5.12) is checked. Through this process, the round consistency tests are correlated to improve soundness. Such a **query test** can be seen as a *global* consistency test, similar to the one of the FRI protocol. For the final test, $\mathscr{V}$ reads $f^{(r)} : \mathcal{P}_r \to \mathbb{F}$ in its entirety to test if $f^{(r)} \in C_r$.

Theorem 5.25. *Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code of length $n \coloneqq |\mathcal{P}_0|$. By definition, C_0 admits a solvable group $\mathcal{G} \in \mathrm{Aut}(\mathcal{X}_0)$ such that $|\mathcal{G}| > n^e$ for a certain $e \in (0,1)$ and induces a sequence of codes (C_i). Denote p_{max} the largest order of the factor groups of the sequence $\bar{\mathcal{G}}$, $\lambda \coloneqq \min_i \Delta(C_i)$ and $\gamma \coloneqq \min\left(J_\varepsilon^{p_{max}}(\lambda), \frac{1}{2}(\lambda + \frac{\varepsilon}{2})\right)$. There is an IOPP system $(\mathscr{P}, \mathscr{V})$ for C_0 satisfying:*

Perfect completeness: *If $f^{(0)} \in C_0$ and $f^{(1)}, \ldots, f^{(r)}$ are honestly generated by the prover, the verifier outputs* accept *with probability 1.*

Soundness: *Assume $f^{(0)}$ is δ-far from C_0 and let $\varepsilon \in (0,1)$. With probability at least $1 - \mathsf{err}_{commit}$ over the randomness of the verifier during the COMMIT phase, where*

$$\mathsf{err}_{commit} \leqslant \frac{\log n}{|\mathbb{F}|}\left(p_{max} + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_{max}}$$

and for any oracles $f^{(1)}, \ldots, f^{(r)}$ adaptively chosen by a possibly dishonest prover $\mathscr{P}^*$, the probability that the verifier $\mathscr{V}$ outputs accept *after a single query test is at most*

$$\mathsf{err}_{query}(\delta) \leqslant (1 - \min(\delta, \gamma) + \varepsilon \log n).$$

Overall, for any prover $\mathscr{P}^*$, the soundness error $\mathsf{err}(\delta)$ after t repetitions of the QUERY phase satisfies

$$\begin{aligned}\mathsf{err}(\delta) &\leqslant \mathsf{err}_{commit} + \left(\mathsf{err}_{query}(\delta)\right)^t \\ &< \frac{\log n}{|\mathbb{F}|}\left(p_{max} + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_{max}} + (1 - \min(\delta, \gamma) + \varepsilon \log n)^t.\end{aligned}$$

Moreover, the IOPP system is public-coin, has round complexity $\mathsf{r}(n) < \log n$, proof length $\mathsf{l}(n) < n$ and query complexity $\mathsf{q}(n) < t p_{max} \log n + n^{1-e}$.

Proof. Lemma 5.21, Proposition 5.22 and Proposition 5.23 satisfy the conditions of Definition 3.1 and Definition 3.4. Completeness and soundness are given by Theorem 3.8. Let us prove the rest of the theorem.

(Round complexity) We have that $\prod_{i=0}^{r-1} p_i = \frac{n}{n_r}$, where $n_r = |\mathcal{P}_r| = \frac{n}{|\mathcal{G}|} < n^{1-e}$. For every $i \in [\![0, r-1]\!]$, $2 \leqslant p_i \leqslant p_{max}$. Therefore $\mathsf{r}(n) \leqslant \log_2 n - \log_2 n_r < \log_2 n$.

(Query complexity) Notice that for $i \in [\![0, r-2]\!]$, $f^{(i+1)}(Q_{i+1})$ is reused for the next round consistency test. Hence, $\mathsf{q}(n) = t\left(\sum_{i=0}^{r-1} p_i\right) + n^{1-e} \leqslant trp_{max} + n^{1-e}$.

(Proof length) The total proof length $\mathsf{l}(n)$ is the sum of the lengths of all the oracles provided by $\mathscr{P}$ during the COMMIT phase, counted in field elements. Denoting $t_{i+1} \coloneqq \prod_{j=0}^{i} p_j$, we notice that $|\mathcal{P}_{i+1}| = \frac{|\mathcal{P}_i|}{p_i} = \frac{|\mathcal{P}_0|}{t_{i+1}}$. Thus, we have

$$\mathsf{l}(n) = \sum_{i=1}^{r} |\mathcal{P}_i| = \sum_{i=1}^{r} \frac{|\mathcal{P}_0|}{t_i} \leqslant n \sum_{i=1}^{r} \frac{1}{2^i} < n.$$

□

5.5 A family of foldable AG codes on Kummer curves

5.5.1 Preliminaries

Let us consider a Kummer curve over a finite field $\mathbb{F}$ defined by an equation of the form

$$\mathcal{X} : Y^N = f(X) = \prod_{\ell=1}^{m} (X - \alpha_\ell) \tag{5.13}$$

where f is a degree-m separable polynomial of $\mathbb{F}[X]$, $\gcd(N, m) = 1$ and $\alpha_\ell \in \mathbb{F}$ for all $\ell \in [\![1, m]\!]$. Let us denote by P_ℓ the point $(\alpha_\ell, 0)$ and P_∞ the unique point of $\mathcal{X}$ lying on the line at infinity.

Sequence of curves. Assume that $\gcd(N, |\mathbb{F}|) = 1$. The group $\mathbb{Z}/N\mathbb{Z}$ acts on $\mathcal{X}$ via the morphism $(x, y) \mapsto (x, \zeta y)$ where ζ is a primitive N^{th} root of unity. We assume that ζ belongs to $\mathbb{F}$.

The cyclic group $\mathbb{Z}/N\mathbb{Z}$ is solvable: writing the prime decomposition of $N = \prod_{i=0}^{r-1} p_i$ gives the following sequence of subgroups

$$\mathbb{Z}/N\mathbb{Z} \rhd \mathbb{Z}/N_1\mathbb{Z} \rhd \mathbb{Z}/N_2\mathbb{Z} \rhd \cdots \rhd \mathbb{Z}/N_{r-1}\mathbb{Z} \rhd 1, \tag{5.14}$$

where

$$N_i \coloneqq \prod_{j=i}^{r-1} p_j. \tag{5.15}$$

The i-th factor group Γ_i is isomorphic to the cyclic group of prime order $\mathbb{Z}/p_i\mathbb{Z}$. It is spanned by $\gamma_i : (x, y) \mapsto (x, \zeta_i y)$ where ζ_i is a primitive p_i^{th} root of unity.

Set $\mathcal{X}_0 \coloneqq \mathcal{X}$. By Section 5.3.1, the composition series (5.14) gives a sequence of curves $(\mathcal{X}_i)$ in which the i^{th} curve is defined by

$$\mathcal{X}_i : Y^{N_i} = f(X) \tag{5.16}$$

and has genus

$$g_i = \frac{(N_i - 1)(m-1)}{2}.$$

The last curve $\mathcal{X}_r$ has genus 0 and is isomorphic to the projective line $\mathbb{P}^1$. These successive quotients provide a sequence of projections $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ defined by $\pi_i(x,y) = (x, y^{p_i})$:

$$\overset{\gamma_0}{\curvearrowright} \mathcal{X}_0 \xrightarrow{\pi_0} \dots \xrightarrow{\pi_i} \overset{\gamma_i}{\curvearrowright} \mathcal{X}_i \xrightarrow{\pi_{i+1}} \overset{\gamma_{i+1}}{\curvearrowright} \mathcal{X}_{i+1} \longrightarrow \dots \xrightarrow{\pi_{r-1}} \mathcal{X}_r \simeq \mathbb{P}^1.$$

Example 5.26. The Hermitian curve defined over $\mathbb{F}_{q^2}$ by

$$\mathcal{X}_0 : Y^{q+1} = X^q + X \tag{5.17}$$

is a well-studied particular case of Kummer type curve. In this case, every curve in a $(\mathcal{X}, \bar{\mathcal{G}})$-sequence is maximal over $\mathbb{F}_{q^2}$, *i.e.* $\left|\mathcal{X}_i(\mathbb{F}_{q^2})\right| = q^2 + 1 + 2g_i q$ [Lac87, Proposition 6].

Stabilized points. Let us denote P_∞^i the unique point at infinity on the curve $\mathcal{X}_i$. One can easily check that

$$P_\infty^i := \begin{cases} (1:0:0) & \text{if } N > m \\ (0:1:0) & \text{otherwise.} \end{cases}$$

The points of $\mathcal{X}_0$ whose stabilizer under $\mathbb{Z}/N\mathbb{Z}$ is non trivial are in fact fixed by $\mathbb{Z}/N\mathbb{Z}$ and consist precisely in $P_1, \dots, P_m$ and P_∞^i.

5.5.2 Decomposition of Riemman-Roch spaces for Kummer extensions

The theory of Kummer extensions provides us a decomposition like (5.4) at each level, with $\mu_i = y$ for every $i \in [\![0, r-1]\!]$.

Theorem 5.27 ([Mah04, Theorem 2.2])**.** *Let $D_i \in \mathrm{Div}(\mathcal{X}_i)$ that is Γ_i-invariant. Then*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{p_i-1} L_{\mathcal{X}_{i+1}}\left(\left\lfloor \frac{1}{p_i}(\pi_i)_*(D_i + j\,\mathrm{div}_{\mathcal{X}_i}(y))\right\rfloor\right).$$

An example of a sequence of y-compatible divisors. In order to exhibit a sequence of divisors (D_i) such that D_{i+1} is (D_i, y)-compatible for every $i \geqslant 0$, we need to handle the divisor associated to y and some other elementary functions on each curve $\mathcal{X}_i$, described for instance in [MQS15].

Lemma 5.28 ([MQS15])**.** *On $\mathcal{X}_i$ for every $i \in [\![0, r-1]\!]$, we have*

1. $\mathrm{div}_{\mathcal{X}_i}(x - \alpha_\ell) = N_i(P_\ell - P_\infty^i)$,
2. $\mathrm{div}_{\mathcal{X}_i}(y) = P_1 + \dots + P_m - mP_\infty^i$.

We now give sufficient conditions on the curve $\mathcal{X}_0$ and the first divisor D_0 to get a sequence of compatible divisors.

Lemma 5.29. *Set $D_0 = \sum_{\ell=1}^{m} a_{0,\ell} P_\ell + b_0 P_\infty^0 \in \mathrm{Div}(\mathcal{X}_0)$. Assume that $m \equiv -1 \mod N$ and that the integers $a_{0,1}, \dots, a_{0,m}, b_0$ are all divisible by N. For every $i \in [\![0, r-1]\!]$, set $D_{i+1} = \frac{D_i}{p_i}$. Then, the divisor D_{i+1} is (D_i, y)-compatible.*

Proof. For $i \in [\![1, r]\!]$, let us set $a_{i,\ell} = \frac{a_{i-1,\ell}}{p_{i-1}}$ and $b_i = \frac{b_{i-1}}{p_{i-1}}$ such that

$$D_i = \sum_{\ell=1}^{m} a_{i,\ell} P_\ell + b_i P_\infty^i.$$

Fix $i \in [\![0, r-1]\!]$. The divisor D_i is supported only by Γ_i-fixed points. For any $j \in [\![0, p_i - 1]\!]$, we have

$$E_{i,j} \coloneqq \left\lfloor \frac{1}{p_i} \pi_{i*}(D_i + j \operatorname{div}_{\mathcal{X}_i}(y)) \right\rfloor = \sum_{\ell=1}^{m} \left\lfloor \frac{a_{i,\ell} + j}{p_i} \right\rfloor P_\ell + \left\lfloor \frac{b_i - jm}{p_i} \right\rfloor P_\infty^{i+1}.$$

Since N_i divides N, we have $m \equiv -1 \mod N_i$. Write $m = \kappa_i N_i - 1$ with $\kappa_i \geqslant 1$. The hypothesis on the integers $a_{0,1}, \ldots, a_{0,m}, b_0$ entails

$$\left\lfloor \frac{a_{i,\ell} + j}{p_i} \right\rfloor = a_{i+1,\ell} + \left\lfloor \frac{j}{p_i} \right\rfloor = a_{i+1,\ell}$$
$$\left\lfloor \frac{b_i - jm}{p_i} \right\rfloor = b_{i+1} - \frac{j\kappa_i N_i}{p_i} + \left\lfloor \frac{j}{p_i} \right\rfloor = b_{i+1} - j\kappa_i N_{i+1}.$$

Then

$$E_{i,j} = D_{i+1} - j\kappa_i N_{i+1} P_\infty^{i+1}.$$

In particular, $D_{i+1} = E_{i,0}$ and $E_{i,j} \leqslant D_{i+1}$. Any $\nu_{i+1,j} \coloneqq (x - \alpha)^{\kappa_i j}$ with $\alpha \in \{\alpha_1, \ldots, \alpha_m\}$ gives the last condition on D_{i+1} for it to be (D_i, y)-compatible by Definition 5.12, i.e. $D_{i+1} - E_{i,j} = \operatorname{div}_\infty(\nu_{i+1,j})$. □

5.5.3 Foldable AG codes on Kummer curves and their parameters

We have gathered all the components to exhibit a foldable code on a family of Kummer curves.

Proposition 5.30. *Let $\mathcal{X}_0$ be a Kummer curve defined by* (5.13) *with $m \equiv -1 \mod N$. Take an evaluation set*

$$\mathcal{P}_0 \subseteq \mathcal{X}_0(\mathbb{F}) \backslash \{P_1, \ldots, P_m, P_\infty^0\}$$

formed by $\mathbb{Z}/N\mathbb{Z}$-orbits and $D_0 \in \operatorname{Div}(\mathcal{X}_0)$ satisfying hypothesis of Lemma 5.29. If $N > n^e$ for some $e \in (0, 1)$, then the AG code $C = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ is foldable.

The length of foldable codes over a Kummer curve as defined in (5.13) over $\mathbb{F}_q$ is bounded from above by $q + 1 + (N-1)(\kappa N - 2)\sqrt{q} - \kappa N$, using Hasse-Weil bound, write $m = \kappa N - 1$.

Remark 5.31.

1. *We assumed that primitive N^{th} root ζ was an element of the base field $\mathbb{F}$ (Section 5.5.1) to ensure that the set $\mathcal{P}_0$ is not empty.*
2. *The condition on the coefficients of D_0 (Lemma 5.29) can be loosen while the previous statement still holds. If $a_{0,1}, \ldots, a_{0,m}, b_0$ are divisible by $\prod_{i=0}^{r-2} p_i$ and not necessarily by p_{r-1}, we choose*

$$a_{r,\ell} = \left\lfloor \frac{a_{r-1,\ell}}{p_{r-1}} \right\rfloor \quad \text{and} \quad b_r = \left\lfloor \frac{b_{r-1}}{p_{r-1}} \right\rfloor$$

for the coefficients of D_r. The last curve $\mathcal{X}_r$ being isomorphic to $\mathbb{P}^1$, the existence of balancing functions is trivial, if the first requirement of Definition 5.12 holds.

Explicit basis of the Riemman-Roch spaces. AG codes from the Kummer curve $\mathcal{X}$ associated to divisors as defined in Lemma 5.29 have been studied by Hu and Yang [HY18]. They provide a basis of the Riemman-Roch spaces in a combinatorial form.

Theorem 5.32 ([HY18, Theorem 5]). *Let $j, j_2, \dots, j_m$ be integers. We define*

$$E_{j,j_2,\dots,j_m} \coloneqq y^j \prod_{\ell=2}^{m} (x-\alpha_\ell)^{j_\ell}.$$

Consider $D = \sum_{\ell=1}^{m} a_\ell P_\ell + bP_\infty$. Set

$$\Omega_{a_1,\dots,a_m,b} \coloneqq \left\{ (j, j_2, \dots, j_m) \mid j + a_1 \geqslant 0,\ j_\ell = \left\lceil \frac{-j-a_\ell}{N} \right\rceil \text{ for } \ell = 2,\dots,m \right.$$
$$\left. \text{and } mj + N(j_2 + \cdots + j_m) \leqslant b \right\}.$$

Then the elements $E_{j,j_2,\dots,j_m}$ for $(j, j_2, \dots, j_m) \in \Omega_{a_1,\dots,a_m,b}$ form a basis of $L_{\mathcal{X}}(D)$.

Parameters. To estimate the parameters of the code by using the Riemnann-Roch theorem, we shall rely on the following result. Recall that $n_i = |\mathcal{P}_i|$.

Lemma 5.33. *Let g_0 by the genus of $\mathcal{X}_0$. Assume that $2(g_0-1) < \deg(D_0)$ (resp. $\deg(D_0) < n_0$). Then for every $i \in [\![0, r]\!]$, $2(g_i - 1) < \deg(D_i)$ (resp. $\deg(D_i) < n_i$).*

Proof. It is enough to notice that for every $i \in [\![0, r-1]\!]$,

$$\deg D_{i+1} = \frac{\deg D_i}{p_i}, \qquad n_{i+1} = \frac{n_i}{p_i}, \qquad \text{and} \quad g_{i+1} \leqslant \frac{g_i}{p_i}.$$

□

In other words, if the degree of the first divisor is such that we can estimate the parameters of C_0 thanks to Riemann-Roch Theorem, then we handle the parameters of all the sequence of codes.

Proposition 5.34. *If $\deg(D_0) < n_0$, then for every $i \in [\![0, r]\!]$, the code C_i has length n_i and minimum relative distance $\Delta(C_i) = 1 - \frac{\deg D_0}{n_0}$. In particular, the RS code C_r has length $\frac{n_0}{N}$, dimension $\frac{\deg D_0}{N} + 1$ and relative minimum distance $1 - \frac{\deg D_0}{n_0}$.*

Moreover, if $2(g_0-1) < \deg(D_0)$, for every $i \in [\![0, r]\!]$, the code C_i has dimension $\deg D_i - g_i + 1$.

Proof. The length of C_i is n_i by construction and its dimension is given by the Riemann-Roch theorem. So let us prove the statement concerning the relative minimum distance.

First notice that $n_i = p_i n_{i+1}$ and $\deg(D_i) = p_i \deg(D_{i+1})$ so $1 - \frac{\deg D_i}{n_i} = 1 - \frac{\deg D_0}{n_0}$. For $i = r$, the code C_r is a Reed-Solomon code of degree $0 \leqslant \deg(D_r) < n_r$ by Lemma 5.33 and has the expected relative minimum distance.

Now assume that $\Delta(C_{i+1})$ equals $1 - \frac{\deg D_0}{n_0}$ and let us prove that so does $\Delta(C_i)$. On the one hand, the divisor D_{i+1} corresponds to $E_{i,0}$. Thus, for every $f \in C_{i+1}$, $f \circ \pi_i \in C_i$. In addition, the weight of $f \circ \pi_i$ in C_i is p_i times the weight of f in C_{i+1}. Since $n_i = p_i n_{i+1}$, we have $\Delta(C_i) \leqslant \Delta(C_{i+1})$. On the other hand, as $\deg(C_i) < n_i$, we have $\Delta(C_i) \geqslant 1 - \frac{\deg D_i}{n_i}$, which concludes the proof. □

5.6 A family of foldable AG codes along the Hermitian tower

5.6.1 Preliminaries

Sequence of curves. We now consider curves along the Hermitian tower $\mathcal{F} = (\mathcal{X}_i)_{i\geqslant 0}$ (Definition 5.8). We have an infinite sequence of curves $(\mathcal{X}_i)_{i\geqslant 0}$ as follows.

$$\ldots \xrightarrow{\pi_{i+1}} \mathcal{X}_i \xrightarrow{\pi_i} \mathcal{X}_{i-1} \xrightarrow{\pi_{i-1}} \ldots \xrightarrow{\pi_1} \mathcal{X}_0 \simeq \mathbb{P}^1.$$

Remark 5.35. *In the context of recursive towers, it is classical to index the curves the other way round compared to the notations used in Section 5.3.*

For $i = 1$, $\mathcal{X}_1$ is the Hermitian curve over $\mathbb{F}_{q^2}$. One can view the curve $\mathcal{X}_i$ embedded in an $(i+1)$-dimensional affine space with variables $(x_0, x_1, \ldots, x_i)$ defined by the equations

$$X_i^q + X_i = X_{i-1}^{q+1} \quad \text{for } i \geqslant 1. \tag{5.18}$$

Let us denote $F_i \coloneqq \mathbb{F}_{q^2}(\mathcal{X}_i)$ and $g_i := g(\mathcal{X}_i)$ the genus of the curve $\mathcal{X}_i$. An explicit formula was given by Pellikaan, Shen and Wee [PSW91, Proposition 4]. We have $g_0 = 0$ and for $i \geqslant 1$,

$$g_i = \frac{1}{2}\left[(q^2-1)\left((q+1)^i - q^i\right) + 1 - q^i\right] = \frac{1}{2}\cdot\left(\sum_{k=1}^{i} q^{i+1}\cdot\left(1+\frac{1}{q}\right)^{k-1} + 1 - (1+q)^i\right). \tag{5.19}$$

For every $i \geqslant 0$, the number of $\mathbb{F}_{q^2}$–rational places in F_i is given by

$$\left|\mathcal{X}_i(\mathbb{F}_{q^2})\right| = q^{i+2} + 1.$$

This tower is a tower of *Artin-Schreier extensions*, which have been extensively studied (see for example [Sti09, Section 3.7]). Let us recall some classical results that will be useful to design foldable AG codes along this tower.

Automorphisms and projection maps. The set $A \coloneqq \left\{\alpha \in \mathbb{F}_{q^2} \mid \alpha^q + \alpha = 0\right\}$ is a subgroup of order q of the additive group of $\mathbb{F}_{q^2}$. Assuming that $\mathbb{F}_q$ is an extension of degree k of the prime field $\mathbb{F}_p$, $p = \operatorname{char} \mathbb{F}_q$, A is isomorphic to $(\mathbb{Z}/p\mathbb{Z})^k$. For any i, let us consider

$$\Gamma_i \coloneqq \{\gamma_{i,\alpha} : (x_1, \ldots, x_i) \mapsto (x_1, \ldots, x_{i-1}, x_i + \alpha) \mid \alpha \in A\}.$$

We have that Γ_i is a subgroup of $\operatorname{Aut}(\mathcal{X}_i)$ of order q and $\Gamma_i \simeq (\mathbb{Z}/p\mathbb{Z})^k$. Moreover, $\mathcal{X}_i/\Gamma_i = \mathcal{X}_{i-1}$. The quotient map $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i-1}$ consists in the projection onto the first i coordinates. For every $i \geqslant 0$, we set Π_i to be the composition of the first i quotient maps, *i.e.*

$$\Pi_i \coloneqq \pi_i \circ \pi_{i-1} \circ \cdots \circ \pi_0.$$

Behaviour of the point of infinity. In what follows, let us denote by $P_\infty^{(0)}$ the unique pole of the function x_0 in F_0, which corresponds to the point at infinity on the projective line $\mathcal{X}_0 = \mathbb{P}^1$.

Lemma 5.36 ([Sti09, Proposition 3.7.8])**.** *Let $i \geqslant 1$. The place $P_\infty^{(0)}$ is totally ramified in F_i, which means that the preimage $\Pi_i^{-1}\left(\left\{P_\infty^{(0)}\right\}\right)$ consists in a unique place, denoted by $P_\infty^{(i)} \in \mathcal{X}_i$. Moreover, $P_\infty^{(0)}$ is the unique place that is ramified in the tower $\mathcal{F}$.*

The peculiar behaviour of the points $P_\infty^{(i)}$ in the tower encourages us to define a sequence of codes associated with divisors $D_i \in \mathrm{Div}(\mathcal{X}_i)$ of the form

$$D_i \coloneqq d_i P_\infty^{(i)} \text{ for } i \geqslant 1.$$

Let us focus on the principal divisors $\mathrm{div}_{\mathcal{X}_i}(x_j)$ $(0 \leqslant j \leqslant i)$ and their valuation at the point $P_\infty^{(i)}$. Their properties follow from the study of the basic function field $F = \mathbb{F}_{q^2}(x,y)$ which is nothing but the Hermitian function field. It is a special case of Artin-Schreier extension of $\mathbb{F}_{q^2}(x)$ and is well-known that we have

$$\mathrm{div}_{\mathcal{X}_1}(y) = P^{(0)} - P_\infty^{(0)}.$$

Remark 5.37. *The role of the variables x and y is reversed compared to the Kummer model of the Hermitian curve, studied in the previous section.*

Since each extension F_i/F_{i-1} corresponds to the same Artin-Schreier extension, and that $P_\infty^{(0)}$ is fully ramified in F_i/F_0, we can deduce the form of the divisor $\mathrm{div}_{\mathcal{X}_i}(x_i)$, given in the next lemma. The valuation of the function $x_j \in F_{i-1}$ at $P_\infty^{(i-1)}$ follows from the extension degrees $[F_{i-1} : F_j] = q^{i-1-j}$.

Lemma 5.38. *The following two assertions hold.*

1. *For $i \geqslant 1$, we have*
$$\mathrm{div}_{\mathcal{X}_i}(x_i) = (q+1)^i \left(P^{(i)} - P_\infty^{(i)}\right),$$
where $P^{(i)}$ is the unique common zero of the functions $x_0, \ldots, x_i$;
2. *Let $i \geqslant 1$. Then for $0 \leqslant j \leqslant i-1$, the valuation of the function $x_j \in F_{i-1}$ is given by*
$$v_{P_\infty^{(i-1)}}(x_j) = -q^{i-1-j}(q+1)^j.$$

Basis of the Riemann-Roch spaces associated to the divisor $d_i P_\infty^{(i)}$. For a given $i \geqslant 0$, $P_\infty^{(i)}$ is the unique pole of the functions $x_0, \ldots, x_i$, which gives an explicit basis of the Riemman-Roch space associated to a multiple of $P_\infty^{(i)}$.

Lemma 5.39. *For all $i \leqslant 1$ and $m \leqslant 1$, the Riemann-Roch $L_{F_i}(mP_\infty^{(i)})$ is formed by linear combinations of functions in the following set:*

$$\left\{ x_0^{a_0} \cdots x_i^{a_i} \mid 0 \leqslant a_0\ ,\ 0 \leqslant a_j \leqslant q-1 \text{ and } \sum_{j=0}^{i} a_j q^{i-j}(q+1)^j \leqslant m \right\}.$$

5.6.2 Decomposition of Riemann-Roch spaces and balancing functions

Let us fix $i \geqslant 0$. We aim to define a sequence of AG codes on the tower of curves $(\mathcal{X}_i)_{i\geqslant 0}$ defined by

$$C(\mathcal{X}_i, \mathcal{P}_i, D_i) \text{ where } \mathcal{P}_i \subseteq \mathcal{X}_i(\mathbb{F}_{q^2})\backslash\{P_\infty^{(i)}\} \text{ and } D_i = d_i P_\infty^{(i)}.$$

In order to obtain a sequence of foldable codes, we need to describe the Riemann-Roch spaces on a certain step from Riemann-Roch spaces on lower curves. Since we consider the action of a group of order q, Kani's theorem does not apply. However, we can find a decomposition by hand. We deduce such a decomposition from the explicit basis of the Riemman-Roch space given in Lemma 5.39.

Proposition 5.40. *Let $i \geqslant 0$. Set $D_i = d_i P_\infty^{(i)}$ for some integer d_i. Then*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{q-1} x_i^j \pi_i^* \left(L_{\mathcal{X}_{i-1}}(E_{i,j})\right)$$

with

$$E_{i,j} \coloneqq \left\lfloor \frac{1}{q} \pi_{i*} \left(D_i - j \cdot \operatorname{div}_{\mathcal{X}_i}(x_i)\right) \right\rfloor \quad \text{for } 0 \leqslant j \leqslant q-1.$$

In other words, the function $x_i \in F_i$ partitions the divisor D_i in the sense of Definition 5.11.

Proof. By Lemma 5.39, $L_{\mathcal{X}_i}(D_i)$ is formed by linear combinations of $x_0^{a_0} \cdots x_i^{a_i}$ with non negative exponents such that $0 \leqslant a_j \leqslant q-1$ for $j \neq 1$ and $\sum_{j=0}^{i} a_j q^{i-j}(q+1)^j \leqslant d_i$. As a_j runs in $\{0, \ldots, q-1\}$, the proof is concluded by noticing that the function $x_0^{a_0} \cdots x_{i-1}^{a_{i-1}} \in F_i$ lies in $L_{X_i}(D_i - j \cdot \operatorname{div}_{\mathcal{X}_i}(x_i))$ which means that $x_0^{a_0} \cdots x_{i-1}^{a_{i-1}} \in F_{i-1}$ belongs to $L_{\mathcal{X}_{i-1}}(E_{i,j})$.

□

To make D_{i-1} compatible with (D_i, x_i) (Definition 5.12), we need the existence of q balancing functions $v_{i-1,j} \in F_{i-1}$ (for every $0 \leqslant j \leqslant q-1$) such that

$$D_{i-1} - E_{i,j} = (v_{i-1,j})_\infty. \tag{5.20}$$

In our setup, we have

$$E_{i,j} = \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor P_\infty^{(i-1)}.$$

Thus, we need to "balance" the divisors

$$D_{i-1} - E_{i,j} = \left(d_{i-1} - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor\right) P_\infty^{(i-1)}.$$

We are led to study the Weierstrass semigroup of $P_\infty^{(i-1)}$, denoted by $H\left(P_\infty^{(i-1)}\right)$. The generators of this semigroup can be found using Lemma 5.38. In fact, $P_\infty^{(i-1)}$ is the unique common pole of the functions $x_0, \ldots, x_{i-1} \in F_{i-1}$ and we know their exact valuation. Thus we have

$$H\left(P_\infty^{(i-1)}\right) = \left\langle q^{i-1-k}(q+1)^k \,,\, 0 \leqslant k \leqslant i-1 \right\rangle_{\mathbb{N}}.$$

Remark 5.41. *In the spirit of the FRI protocol, one could be tempted to choose D_{i-1} as $E_{i,0}$. Such a choice would be valid in the sense of Definition 5.12 if and only for every $0 \leqslant j \leqslant q-1$*

$$\left\lfloor \frac{d_i}{q} \right\rfloor - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor \in H\left(P_\infty^{(i-1)}\right).$$

Unfortunately, when i increases, this condition is never satisfied.

To ensure that $\deg(D_{i-1} - E_{i,j})$ is never a Weierstrass gap for $P_\infty^{(i-1)}$, we increase the degree d_{i-1} of D_{i-1}.

Theorem 5.42. *Let $i \geqslant 1$. Set $D_i = d_i P_\infty^{(i)}$ for some integer d_i and $D_{i-1} = d_{i-1} P_\infty^{(i-1)}$ where*

$$d_{i-1} \coloneqq \left\lfloor \frac{d_i}{q} \right\rfloor + 2g_{i-1}. \tag{5.21}$$

Then D_{i-1} is compatible with (D_i, x_i) (Definition 5.12).

Proof. By Theorem 5.9, we know that

$$\max\left(\mathbb{N} \backslash H\left(P_\infty^{(i-1)}\right)\right) \leqslant 2g_{i-1} - 1.$$

Then for every $0 \leqslant j \leqslant q-1$, the difference

$$m_{i,j} \coloneqq \deg(D_{i-1} - E_{i,j}) = \left(\left\lfloor \frac{d_i}{q} \right\rfloor - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor + 2g_{i-1}\right) \tag{5.22}$$

always belongs to the Weierstrass semigroup $H\left(P_\infty^{(i-1)}\right)$. □

About the balancing functions. Since we know a $\mathbb{N}$-basis of the Weierstrass semigroup at $P_\infty^{(i-1)}$, we are able to explicit the form of the functions $\nu_{i-1,j}$. In particular, they can be chosen as product of powers of the functions $x_0, \ldots, x_{i-1}$. More precisely, if $a_{i,j} \coloneqq (a_{i,j}(0), \ldots, a_{i,j}(i-1)) \in \mathbb{N}^i$ are integers such that

$$m_{i,j} = \sum_{k=0}^{i-1} a_{i,j}(k) \cdot q^{i-1-k}(q+1)^k, \tag{5.23}$$

then $m_{i,j} \in H\left(P_\infty^{(i-1)}\right)$. The corresponding choice for the balancing function is then given by

$$\nu_{i-1,j} = \prod_{k=0}^{i-1} x_k^{a_{i,j}(k)}.$$

Note that finding a vector $a_{i,j} \in \mathbb{N}^i$ satisfying (5.23) leads to the study of the diophantine equation

$$m_{i,j} = \sum_{k=0}^{i-1} a_k \cdot q^{i-1-k}(q+1)^k$$

with i unknowns $a_k \in \mathbb{N}$, for which we know there exists at least a solution (and we only need one).

5.6.3 Foldable AG codes along the Hermitian tower

We denote by $i_{\max}$ the level in the tower $(\mathcal{X}_i)_{i\geqslant 0}$, such that $\mathcal{X}_{i_{\max}}$ is the curve on which the code we want to test proximity is defined. Theorem 5.42 yields the following proposition.

Proposition 5.43. *Fix an integer $i_{\max}$. Set $\mathcal{P}_0 \subseteq \mathbb{P}^1(\mathbb{F}_{q^2})\backslash\{P_\infty^0\}$ and define $\mathcal{P}_{i_{\max}}$ as the preimage of $\mathcal{P}_0$ under the morphism $\Pi_{i_{\max}}$. Fix an integer $d_{i_{\max}}$. Then the code $C(\mathcal{X}_{i_{\max}}, \mathcal{P}_{i_{\max}}, d_{i_{\max}} P_\infty^{(i)})$ is foldable.*

To control the dimension of foldable codes, we will focus on those of the form

$$C \coloneqq C\left(\mathcal{X}_{i_{\max}}, \mathcal{X}_{i_{\max}}(\mathbb{F}_{q^2})\backslash\left\{P_\infty^{(i_{\max})}\right\}, (2\alpha+1)g_{i_{\max}})P_\infty^{(i_{\max})}\right) \tag{5.24}$$

for some $\alpha > 1/2$. In this case, we have $n_{i_{\max}} = q^{i_{\max}+2}$. We can determine a sufficient condition over $i_{\max}$ and α to get a family of codes with constant rate and polylogarithmic-size alphabet.

To estimate the parameters of the foldable codes we define along the Hermitian tower, we need to handle the genera of the curves in this tower.

Proposition 5.44. *For $i \geqslant 1$, we have*

$$g_i \leqslant \frac{q^{i+1}}{2}\sum_{k=1}^{i}\binom{i}{k}\frac{1}{q^{k-1}} \leqslant \frac{iq^{i+1}}{2}\sum_{k=1}^{i}\left(\frac{i}{q}\right)^{k-1} \leqslant \frac{i}{2}q^{i+1} + \frac{i(i-1)}{2}q^i,$$

the last inequality holding only if $2(i-1) < q$.

Proof. Starting from the second formula of (5.19), we can write

$$g_i = \frac{1}{2}\cdot\left(q^{i+1}\sum_{k=1}^{i}\left(1+\frac{1}{q}\right)^{k-1} + 1 - (1+q)^i\right) \leqslant \frac{q^{i+1}}{2}\cdot q\cdot((1+1/q)^i - 1) = \frac{q^{i+1}}{2}\cdot\sum_{k=1}^{i}\binom{i}{k}\frac{1}{q^{k-1}},$$

using that the term outside the geometric sum is non positive. Note that if $k \geqslant 2$, then we can bound the binomials coefficients as follows

$$\binom{i}{k} = \frac{i(i-1)\cdots(i-k+1)}{k(k-1)\cdots 2} \leqslant \frac{i(i-1)^{k-1}}{2},$$

as the denominator is greater than 2 and the factors $i-1, i-2, ..., i-k+1$ are all lesser than $i-1$. Factoring and using this upperbound over the binomials coefficients, we get

$$g_i \leqslant \frac{q^{i+1}}{2}\cdot\left(i + \frac{i}{2}\sum_{k=2}^{i}\left(\frac{i-1}{q}\right)^{k-1}\right) = \frac{iq^{i+1}}{2}\left(1 + \frac{1}{2}\cdot\left(\frac{i-1}{q}\right)\cdot\sum_{k=2}^{i}\left(\frac{i-1}{q}\right)^{k-2}\right).$$

Assuming that $2(i-1) < q$, we can bound the sum $\sum_{k=2}^{i}\left(\frac{i-1}{q}\right)^{k-2}$ by 2, which concludes the proof. □

Lemma 5.45. *Fix $\varepsilon \in (0,1)$ and set $i = q^\varepsilon$. Then*

$$\frac{g_i}{q^{i+2}} \underset{q\to\infty}{\sim} \frac{1}{2q^{1-\varepsilon}}.$$

Proof. From the first formula of (5.19), we get

$$\frac{2gi_{\max}}{q^{i_{\max}+2}} = \left(1-\frac{1}{q^2}\right)\left[\left(1+\frac{1}{q}\right)^{q^\varepsilon}-1\right]+\frac{1}{q^{i_{\max}+2}}-\frac{1}{q^2}$$

Let us examine the asymptotic behaviour of $\left(1+\frac{1}{q}\right)^{q^\varepsilon}$ when q goes to infinity. Set $h=q^{-1}$.

$$\left(1+\frac{1}{q}\right)^{q^\varepsilon} = \exp\left(h^{1-\varepsilon}\cdot\frac{\log(1+h)}{h}\right) = \exp\left(h^{1-\varepsilon}\left(1-\frac{h}{2}+o(h)\right)\right) = 1+h^{1-\varepsilon}+o(h^{1-\varepsilon})$$

Therefore, we have

$$\left(1+\frac{1}{q}\right)^{q^\varepsilon}-1 \sim \frac{1}{q^{1-\varepsilon}}.$$

□

Lemma 5.46. *Let $R\in(0,1)$. Fix $\varepsilon\in(0,1)$. Set $i_{\max}\coloneqq q^\varepsilon$ and $\alpha\coloneqq Rq^{1-\varepsilon}$. The ratio of the dimension of the code C defined in* (5.24) *by its block length goes to R when q tends to infinity. Moreover, if $2(q^\varepsilon-1)<q$, the relative minimum distance of C is bounded from below by $1-R\left(1+\frac{1}{q}\right)$.*

Proof. If $\alpha>\frac{1}{2}$, the dimension of the code is equal to $(2\alpha+1)g_{i_{\max}}-g_{i_{\max}}+1 = 2Rq^{1-\varepsilon}g_{i_{\max}}+1$ by the Riemann-Roch Theorem. As R is fixed and q goes to infinity, we can assume that $\alpha>1/2$ to compute the rate as

$$\lim_{q\to\infty}\frac{2Rq^{1-\varepsilon}g_{i_{\max}}}{q^{i_{\max}+2}}$$

for $i_{\max}=q^\varepsilon$. Lemma 5.45 clearly implies that this limit is equal to R.

Regarding the relative minimum distance, we use the Goppa bound: if $(2\alpha+1)g_{i_{\max}}<q^{i_{\max}+2}$, then the relative minimum distance of C satisfies $\Delta(C)\geqslant 1-\frac{(2\alpha+1)g_{i_{\max}}}{q^{i_{\max}+2}}$. By Proposition 5.44, we have

$$\frac{g_{i_{\max}}}{q^{i_{\max}+2}}\leqslant\frac{i_{\max}}{2q}\left(1+\frac{i_{\max}}{q}\right),$$

which gives the expected lowerbound for our choice of α and $i_{\max}$. □

Given a foldable code as in Proposition 5.43, our AG-IOPP (Section 5.4) involves a sequence of codes (C_i) where each code C_i is as follows:

$$C_i\coloneqq C(\mathcal{X}_{i_{\max}-i},\mathcal{P}_{i_{\max}-i},D_{i_{\max}-i}) \text{ where } \mathcal{P}_{i-1}=\pi_i(\mathcal{P}_i) \text{ and } D_i=d_iP_\infty^{(i)}$$

with the integers d_i defined recursively by

$$d_{i-1}\coloneqq\left\lfloor\frac{d_i}{q}\right\rfloor+2g(\mathcal{X}_{i-1}).$$

Unlike the Kummer case, we have to increase the degree of the divisor by twice the genus of the curve at each step to make sure the compatibility hypotheses of Definition 5.12 are valid. This has a counterpart: the dimension of the codes C_i decreases much slowly than their block length. A foldable code in the sense of Definition 5.13 may induce a sequence of codes in which the last code $C_{i_{\max}}$ is trivial. In this case, the protocol would no longer be sound. We thus need to control the dimension of the code $C_{i_{\max}}$.

Bounding the rate of the underlying Reed-Solomon code. We aim to bound the dimension of the code Reed-Solomon code $C_{i_{\max}}$. Let us compute the degree $d_{i_{\max}}$ of the divisor $D_{i_{\max}}$ on $\mathbb{P}^1$.

Lemma 5.47. *For $1 \leqslant j \leqslant i_{\max}$, we have*

$$d_{i_{\max}-j} \leqslant \left\lfloor \frac{d_{i_{\max}}}{q^j} \right\rfloor + \sum_{k=1}^{j} \left\lfloor \frac{2g_{i_{\max}-k}}{q^{j-k}} \right\rfloor + (j-1).$$

Proof. It follows from the definition of the degrees d_i given in (5.21) and by induction on j. □

Using Lemma 5.47 which bounds the genera g_i for $i \leqslant i_{\max}$, we can get an upperbound on d_0.

Corollary 5.48. *Let us assume that $2(i_{\max}-1) < q$. The degree d_0 of the divisor D_0 on $\mathbb{P}^1$ is bounded from above by*

$$d_0 \leqslant \left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6} \cdot (3q - 4 + 2i_{\max})\right).$$

Proof. By Lemma 5.47, we have the following bound over d_0:

$$d_0 \leqslant \left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + \sum_{i=0}^{i_{\max}-1} \left\lfloor \frac{2g_i}{q^i} \right\rfloor + i_{\max} - 1.$$

It is thus enough to estimate the sum $\sum_{k=0}^{i_{\max}-1} \left\lfloor \frac{2g_k}{q^k} \right\rfloor$. By Proposition 5.44,

$$\begin{aligned}\sum_{k=0}^{i_{\max}-1} \left\lfloor \frac{2g_k}{q^k} \right\rfloor &\leqslant \sum_{k=0}^{i_{\max}-1} (kq + k(k-1)) \\ &= (q-1) \cdot \frac{i_{\max}(i_{\max}-1)}{2} + \frac{i_{\max}(i_{\max}-1)(2i_{\max}-1)}{6} \\ &= \frac{i_{\max}(i_{\max}-1)}{2} \cdot \left(q - \frac{4}{3} + \frac{2i_{\max}}{3}\right),\end{aligned}$$

which gives the expected result. □

Depending on the length of the code $C_{i_{\max}}$, we can determine a sufficient condition on $i_{\max}$ that ensures that the code C_0 is not trivial. Let us denote by n_0 the size of $\mathcal{P}_0$. It satisfies $n_0 \leqslant q^2$. The rate of C_0 is equal to $\dfrac{d_0+1}{n_0}$. Also we have $n_{i_{\max}} := |\mathcal{P}_{i_{\max}}| = q^{i_{\max}} n_0$.

Corollary 5.49. *Let us fix $\rho \in (0,1)$. If*

$$\left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6} \cdot (3q - 4 + 2i_{\max})\right) + 1 < \rho n_0,$$

then the rate of the code C_0 is less than ρ.

Foldable codes with constant rate which are endowed with an IOPP with designed soundness. Let us consider a foldable code of the form

$$C = C_0 = C\left(\mathcal{X}_{i_{\max}}, \mathcal{X}_{i_{\max}}(\mathbb{F}_{q^2}) \setminus \left\{P_\infty^{(i_{\max})}\right\}, (2\alpha+1)g_{i_{\max}} P_\infty^{(i_{\max})}\right) \tag{5.24}$$

for some $\alpha > 1/2$, as in Section 5.6.3. The evaluation set $\mathcal{P}_{i_{\max}}$ is the whole set of rational points of $\mathcal{X}_{i_{\max}}$ minus the point of at infinity $P_\infty^{(i_{\max})}$, *i.e.* $n_{i_{\max}} = q^{i_{\max}+2}$.

Proposition 5.50. *Let us fix $\rho \in (0,1)$. The rate of the RS code $C_{i_{\max}}$ below C_0 is less than ρ if*

$$2i_{\max}^3 + 3i_{\max}^2(2\alpha + q - 1) + i_{\max}(6\alpha(q-1) + 7) - 6\rho q^2 < 0.$$

Proof. For large enough value of q, we can assume that $2i_{\max} - 1 < q$. Using Proposition 5.44, we get an upper bound for $d_{i_{\max}}$:

$$d_{i_{\max}} \leqslant (2\alpha+1)\frac{i_{\max}}{2} q^{i_{\max}}(q + (i_{\max} - 1)).$$

From Corollary 5.49, a sufficient condition for the underlying RS code to have a rate less than ρ is

$$(2\alpha+1)\frac{i_{\max}}{2}(q + (i_{\max}-1)) + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6}(3q - 4 + 2i_{\max})\right) + 1 < \rho q^2.$$

Multiplying the inequality by 6, expanding and simplifying gives the condition.

□

Now assume that $i_{\max} = q^\varepsilon$ for $\varepsilon \in (0,1)$. In the constant rate regime described in Lemma 5.46, we have $\alpha i_{\max} = Rq$. The condition above becomes

$$2i_{\max}^3 + 3i_{\max}^2(q-1) + i_{\max}(6Rq + 7) < 6q^2\left(\rho - R\left(1 - \frac{1}{q}\right)\right).$$

If $R\left(1 - \frac{1}{q}\right) < \rho$, the right handside is positive. Let us give a rough estimation of the largest ε such that $i_{\max} = q^\varepsilon$ satisfies this inequality. The left handside being equivalent to $3q^{1+2\varepsilon}$, we have

$$\varepsilon \simeq \frac{1}{2}(1 + \log_q\left(\rho - R\left(1 - \frac{1}{q}\right)\right).$$

Table 5.1 displays some examples of level $i_{\max}$ and initial rate R of foldable codes for which the AG-IOPP reduce the proximity test to testing RS codes of rate ρ. In terms of the soundness of the protocol, it means that λ as defined in Theorem 5.25 is greater than $1 - \rho$.

q	$i_{\max}$	n	R	$1-\rho >$
2^4	3	2^{20}	$1/8$	$1/3$
2^5	5	2^{35}		
2^4	4	2^{24}	$1/16$	$1/3$
2^5	3	2^{25}		$3/4$
	5	2^{35}		$1/2$
2^6	4	2^{36}		$3/4$
	5	2^{42}		$2/3$
	7	2^{54}		$1/2$
2^4	3	2^{20}	$1/32$	$1/2$

Table 5.1: Examples of parameters of foldable codes of rate R along the Hermitian tower. Alphabet is $\mathbb{F}_q^2$ and block length is n. The last column gives a bound on the minimal distance of the RS code.

5.7 Proximity tests for AG codes on Kummer curves and Hermitian towers

When we instantiate the AG-IOPP proposed in Section 5.4.2 for the setting of Kummer curves (Section 5.5) and curves in the Hermitian tower (Section 5.6), we end up with a membership test to a RS code. An RS code is itself a foldable AG code (see Section 5.3.3). In order to lower verifier complexity, we can extend the AG-IOPP by replacing the final test by an IOPP for RS code. The main properties of such an enhanced AG-IOPP is examined in this section.

5.7.1 How to iterate the folding to reach a code of dimension 1

We consider a sequence of foldable AG codes $(C_i)_{0\leqslant i\leqslant s}$ as provided by Section 5.5 (Kummer curves) or Section 5.6 (tower of Hermitian curves). The code $C_s = C(\mathbb{P}^1, \mathcal{P}_s, D_s)$ corresponds to a Reed-Solomon code $\mathrm{RS}[\mathbb{F}, \mathcal{P}_s, d] = \{f : \mathcal{P}_s \to \mathbb{F}; \deg f \leqslant d\}$, where the degree bound depends on the parameters of the code C_0. Taking this into consideration, we want to iterate the folding operation until we get a RS code of dimension 1, as it is done in the FRI protocol [BBHR18a].

As in Example 5.3.3, we set $d_0 = d$ and define $d_{i+1} = \left\lfloor \frac{d_i}{2} \right\rfloor$ for any integer i. Set s' the smallest integer such that $d_{s'} = 0$. Then, we consider the sequence of Reed-Solomon codes $(C_{s+i})_{1\leqslant i\leqslant s'}$ obtained from applying the construction described in Section 5.3 to the initial code C_s. Letting $r = s + s'$, we iteratively reduce the proximity test to the code C_0 to a membership test to the code C_r, which is a Reed-Solomon code of dimension 1. If $f^{(0)} \in C_0$, then $f^{(r)}$ is expected to be a constant function, and this can be tested in a trivial way. We can leverage the fact that C_r is a Reed-Solomon code to extend the protocol described in Section 5.4.2. From Construction 3.5, we obtain a r-rounds IOPP system $(\mathscr{P}, \mathscr{V})$ for C_0.

5.7.2 Properties of the AG-IOPP with Kummer curves

Assume $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ is a foldable AG code of blocklength $n_0 = |\mathcal{P}_0|$ on a Kummer curve $\mathcal{X}_0$ (cf. Proposition 5.30). This means that $\mathcal{X}_0$ is defined by an equation $Y^N = f(X)$, where $f \in \mathbb{F}[X]$ is a separable degree-m polynomial, $m \equiv -1 \mod N$, N is coprime with $|\mathbb{F}|$, $|\mathcal{P}_0| = \alpha N$ for some integer α, and $\deg D_0 < \alpha N$. Assume α is a power of 2 and N is a η-smooth integer for a small fixed parameter $\eta \in \mathbb{N}$.

Proposition 5.34 states that the relative minimum distances of the codes C_i are all equal to $\Delta(C_0) = 1 - \frac{\deg D_0}{\alpha N}$. Therefore, the ordering on the integers involved in the prime decomposition $\prod_{i=0}^{s-1} p_i$ of N does not impact the parameters of the protocol. Moreover, the code $C_s = C(\mathcal{X}_s, \mathcal{P}_s, D_s)$ corresponds to a RS code

$$C_s = \mathsf{RS}\left[\mathbb{F}, \mathcal{P}_s, \frac{\deg D_0}{N}\right] = \left\{ f : \mathcal{P}_s \to \mathbb{F}; \deg f \leqslant \frac{\deg D_0}{N} \right\}$$

of blocklength $|\mathcal{P}_s| = \alpha$, which is itself a foldable AG code (see Example 5.3.3).

Theorem 5.51 (Kummer case)**.** *Let $C = (\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code on a Kummer curve satisfying the hypotheses of Proposition 5.30 with N a η-smooth integer. Denote $n = |\mathcal{P}_0|$. There is an IOPP $(\mathscr{P}, \mathscr{V})$ for C with perfect completeness and soundness as stated in Theorem 5.25. Moreover, for t repetitions of the QUERY phase, we have:*

- *rounds complexity* $\mathsf{r}(n) < \log n$,
- *proof length* $\mathsf{l}(n) < n$,
- *query complexity* $\mathsf{q}(n) \leqslant t\eta \log_2 n + 1$,
- *prover complexity* $\mathsf{tp}(n) = O_\eta(n)$,
- *verifier complexity* $\mathsf{tv}(n) = O_\eta(t \log n)$.

Proof. Noticing that the round complexity is now $\mathsf{r}(n) = s + s'$, straightforward calculations show that the bounds on query complexity and proof length computed in the proof of Theorem 5.25 still hold (assuming that we apply the variant of Construction 3.5 described in Remark 3.7). We estimate prover complexity and verifier complexity below.

(Prover complexity) Fix a round index $i < r - 1$. The balancing functions $\nu_{i+1,j} : \mathcal{P}_{i+1} \to \mathbb{F}$ can be precomputed since they do not depend on $f^{(i)}, z^{(i)}$ (see Remark 5.52). To simplify notation, denote $f = f^{(i)}$. For any $z = (z_1, z_2) \in \mathbb{F}^2$, computing the successive powers $(z_1^j, z_2^j)_{0 \leqslant j < p_i}$ takes $2(p_i - 2)$ multiplications. For each $P \in \mathcal{P}_{i+1}$, an honest prover must compute the coefficients $(a_{j,P})_{0 \leqslant j, <P}$ of the polynomial $I_{f,P}(X)$ of degree $\deg I_{f,P} < p_i$ from the interpolation set $\{(\mu_i(\hat{P}), f(\hat{P})) \mid \hat{P} \in S_P\}$ of size p_i. Notice that $\mu_i = y$, so computing $\mu_i(\hat{P})$ for $\hat{P} \in S_P$ is done for free. Univariate interpolation for a polynomial of degree $< p_i$ can be done in $O(p_i^2)$ by Lagrange interpolation. Overall, one can honestly evaluate $\textbf{Fold}\,[f, z] : \mathcal{P}_{i+1} \to \mathbb{F}$ with $|\mathcal{P}_{i+1}|\, O(p_i^2)$ operations in $\mathbb{F}$. We showed previously that $\sum_{i=1}^{r-1} |\mathcal{P}_i| < n$, thus when summing over $r - 1$ rounds, we get that the cost of (honestly) generating the oracles $f^{(1)}, \ldots, f^{(r-1)}$ is $O_\eta(n)$.

(Verifier decision complexity) Verifier complexity is inferred from the previous discussion about prover complexity. For each round, the verifier computes the successive powers of z_1 and z_2, interpolates $I_{f,P}$ for a point $P \in \mathcal{P}_{i+1}$ in $O(p_i^2)$ operations, then computes $\textbf{Fold}\,[f, z]\,(P)$ in a number of operations which is independent of n. Hence, verifier complexity for repetition parameter t is $\mathsf{t_v}(n) = O_\eta(t \log(n))$.

□

Remark 5.52. *We give the cost of precomputing the evaluation tables of the balancing functions. Letting $\nu_{i+1,j}$ be as defined in proof of Lemma 5.29, the sequence of functions $(\nu_{i+1,j})_{0<j<p_i}$ can be evaluated at the same point $P \in \mathcal{P}_{i+1}$ in time $O(\log m + p_i)$ using exponentiation by squaring. Thus, the evaluations of $\nu_{i+1,1}, \dots \nu_{i+1,p_i-1}$ on $\mathcal{P}_{i+1}$ are obtained with $O((\log m + p_i)\,|\mathcal{P}_{i+1}|)$ operations.*

We give an example of an AG code over a Kummer curve where $p_{max} = 2$.

Example 5.53. On $\mathbb{F}_{q^2}$ with $q = 2^{61} - 1$ (9^{th} Mersenne prime), we consider the curve

$$\mathcal{X}_0 : Y^N = X^3 + X$$

where $N = 2^r$ with $r = 16$. It is maximal [TT14] of genus $g = N - 1$. We consider the code C_0 associated to $D_0 = 2^{17} P_\infty^0$ on an evaluation set $\mathcal{P}_0 \subset \mathcal{X}_0(\mathbb{F}_{q^2})$ of size $n = 2^{20}$. Its dimension equals $\dim C_0 = 2^{16} + 2$ and its relative minimum distance λ is bounded from below by $1 - 2^{-3}$. Take $\varepsilon = 2^{-6.55}$. By Theorem 5.25,

$$\mathsf{err}_{commit} \leqslant \frac{\log(n)}{\left|\mathbb{F}_{q^2}\right|}\left(1 + \frac{4}{\varepsilon}\right)\left(\frac{4}{\varepsilon}\right)^2 \approx 2^{4.33+6+3\cdot 6.55-121} \leqslant 2^{-91}$$

$$\mathsf{err}_{query}(\delta) \leqslant (1 - \delta + \varepsilon \log(n))$$

where $1 - \delta = (1 - \lambda + \varepsilon)^{\frac{1}{3}} \leqslant 0,51384$. Hence

$$\mathsf{err}_{query}(\delta) \leqslant 0,51384 + \frac{20}{2^{6.55}} \approx 0,72728.$$

By running the QUERY phase with repetition parameter $t \geqslant 199$, we get $(\mathsf{err}_{query})^t \leqslant 2^{-91}$ and $\mathsf{err}(\delta) \leqslant 2^{-90}$. The last code C_r is a small Reed-Solomon code of length $n_r = 2^4$ and dimension 2. The total number of rounds of the IOPP is thus $r + 1$.

5.7.3 Properties of the AG-IOPP with towers of Hermitian curves

For foldable AG codes, we rely again on the discussion of Section 5.7.1. This yields the following theorem.

Theorem 5.54. *Let $C = (\mathcal{X}, \mathcal{P}, D)$ be a foldable AG code with alphabet $\mathbb{F} = \mathbb{F}_{q^2}$ on a tower of Hermitian curves satisfying the hypotheses of Proposition 5.43. Letting e be the index of the curve $\mathcal{X}$ in the Hermitian tower $(\mathcal{X}_i)_{i\geqslant 0}$, the length $n = |\mathcal{P}|$ of C is at most q^{e+2}. There is an IOPP $(\mathscr{P}, \mathscr{V})$ with perfect completeness, and soundness as stated in Theorem 5.25. Moreover, for t repetitions of the QUERY phase, we have:*

- *rounds complexity* $\mathsf{r}(n) < \log n$,
- *proof length* $\mathsf{l}(n) < n$,
- *query complexity* $\mathsf{q}(n) \leqslant tq \log n + 1$,
- *prover complexity* $\mathsf{tp}(n) = O(n \cdot \mathsf{M}_{\mathbb{F}}(q) \log(q))$,
- *verifier complexity* $\mathsf{tv}(n) = O(t \log n \cdot \mathsf{M}_{\mathbb{F}}(q) \log(q))$.

Proof. The proof follows from proof of Theorem 5.51, replacing η by q. Prover and verifier complexities are computed from the cost of computing the coefficients of a univariate polynomial of degree less than q from its evaluation on points forming an arithmetic progression in $\mathbb{F}_{q^2}$. This interpolation task can be done in $\mathsf{M}_{\mathbb{F}}(q)\log q + O(\mathsf{M}_{\mathbb{F}}(q))$ base field operations [BS05], where $\mathsf{M}_{\mathbb{F}}(d)$ denotes the cost of multiplying two degree-d univariate polynomials over $\mathbb{F}$. □

Conclusion

Many constructions of succinct non-interactive arguments (SNARG) rely on univariate polynomials for arithmetization. This is due to appealing properties of Reed-Solomon codes: they can have constant rate and constant distance, admit quasilinear-time encoding algorithms and feature a multiplication property that facilitates arithmetization (a component wise product of Reed-Solomon codewords is a codeword of a Reed-Solomon code). While some families of multivariate polynomial codes and algebraic geometry codes present the same features, Reed-Solomon codes have been favored in part because of the existence of a very efficient proximity test for them [BBHR18a] and the lack of equivalent solutions for other families of algebraic codes. Building upon the work of [BBHR18a], this thesis provides proximity tests for multivariate polynomial codes in Chapter 4 and algebraic geometry codes Chapter 5, with efficiency parameters that are similar (or equal) to the FRI protocol for Reed-Solomon proximity testing. Thus, this study covers most important basic algebraic codes.

A prospect of improvement is the soundness error of our proximity tests, which is closely related to "worst-case to average-case reductions for the distance to a code" [BKS18]. Obtaining tight results about the distance to any linear space of an element randomly sampled from an affine space is an important open problem which is also involved in distributed storage and cryptographic protocols (examples can be found in [BCI+20]). In Section 3.2.1, we gave a result about distance preservation for linear combinations whose coefficients are obtained as the output of a basic small-bias generator which may be of independent interest. As mentioned there, relating the bias of the generator with the average distance to linear spaces of linear combinations whose coefficients are output of some small-biased generators remains an intriguing problem, with applications to succinct non-interactive arguments [BCL20].

In [BCI+20], the authors made notable improvements for the specific case where the aforementioned linear space is a Reed-Solomon code. Their analysis relies on algebraic decoding algorithms for Reed-Solomon codes whose alphabet is a rational function field.

For starters, applying the same techniques for multivariate codes would require a global decoding algorithm of multivariate codes that could be extended to the field of rational functions, that we are not aware of. It remains an open question whether an analysis of algebraic decoding algorithms of multivariate codes over the field of rational functions is possible.

Regarding algebraic geometry codes, the proof techniques of [BCI+20] could most certainly be used (algebraic decoding algorithms for algebraic geometry codes are very similar to their analogues for Reed-Solomon codes), but this would yield results that are relevant only for codes over quadratic size field, which is not the interesting context for algebraic geometry codes.

Another way to improve soundness error is to slightly modify the proximity tests themselves. For instance, [BGKS20] proposed a variant of the FRI protocol (named "DEEP-FRI") which relates the list size and radius of the list decoding of the code with the soundness error. Before the work of [BCI+20], the DEEP-FRI protocol induced shorter non-interactive arguments than the FRI protocol. In [BGKS20], the authors left for future work the generalization of their techniques to multivariate codes and algebraic geometry codes. Now that a FRI-like IOPP exists for algebraic geometry codes, the possibility of generalizing the DEEP-FRI protocol to AG codes seems quite plausible, and is left for future work. Regarding tensor product of Reed-Solomon codes, early work that we have conducted indicates that a DEEP-FRI-like variant would reduce the total query complexity for constant sound-

ness, at the cost of a logarithmic overhead in proof length (*e.g.* by relying on the work of [GGR09] about list decoding of tensor products).

In Chapter 3, we formulated a generic abstract framework for constructing efficient interactive oracle proofs of proximity for linear codes, based on the design principles introduced in the FRI protocol. The analysis of this framework has the main benefit of not relying on the underlying algebraic properties of the considered codes, and can conceivably be applied to non-algebraic constructions of codes. Indeed, a recent line of research avoids the use of polynomial codes in IOP constructions and consider tensor product of linear-time encodable codes [BCG20, BCL20, RR20, RR21, GLS+21]. In particular, [BCG20, BCL20] constructed an IOPP for tensor products of *any* linear codes, based on a folding operation and inspired from [BBHR18a].

Finally, this study focuses on the proximity testing part of IOP constructions, letting aside arithmetization techniques. Thus, this leaves open the question of whether our proximity tests can be used to construct practical succinct non-interactive arguments. Proposing highly-efficient IOPPs for codes beyond Reed-Solomon codes might open up a range of different arithmetization techniques for designing explicit constructions of proof systems, with the potential of improving concrete efficiency.

In that direction, we can think of several appealing features of our results. First, FFTs properties enable more efficient encoding of codewords of tensor product of Reed-Solomon codes, compared to a single Reed-Solomon code of the same block length. Since the task of the IOP prover during arithmetization is mainly to encode messages, this may concretely improve the prover running time. Besides, enabling proximity testing for Reed-Muller codes with support L^m where $|L|$ is much smaller than the field size may give more flexibility in the design of proof systems. As for algebraic geometry codes, the possibility of significantly reduce the field size may overall improve the bit complexities of proof systems based on AG codes (compared to Reed-Solomon-based ones), by speeding up field operations. Finally, our IOPP for algebraic geometry codes along the Hermitian tower does not make any assumption on the base field $\mathbb{F}_q$, except that q should be a square. In particular, the field is not required to admit subgroup of large smooth order (which is a requirement of IOPPs for polynomial codes). Such a flexibility is particularly interesting for applications which operate over a predefined finite field.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. (Cited on page 3.)

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104. ACM, 2017. (Cited on pages viii and 14.)

[AJMR19] Vikraman Arvind, Pushkar S. Joglekar, Partha Mukhopadhyay, and S. Raja. Randomized Polynomial-Time Identity Testing for Noncommutative Circuits. *Theory of Computing*, 15(7):1–36, 2019. (Cited on page 13.)

[AKK+03] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Low-Degree Polynomials over GF(2). In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, volume 2764 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2003. (Cited on page 54.)

[AKK+05] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Trans. Inf. Theory*, 51(11):4032–4039, 2005. (Cited on page 54.)

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. extended version of FOCS'92. (Cited on pages v, vii, 4, 5, 8, 52 and 54.)

[Alo99] Noga Alon. Combinatorial Nullstellensatz. *Combinatorics, Probability and Computing*, 8(1–2), 1999. (Cited on page 14.)

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs; A New Characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 2–13. IEEE Computer Society, 1992. (Cited on pages v, vii, 4, 5, 8, 17, 18 and 81.)

[AS03] Sanjeev Arora and Madhu Sudan. Improved Low-Degree Testing and its Applications. *Combinatorica*, 23(3):365–426, 2003. (Cited on pages 53 and 54.)

[Bab85] László Babai. Trading Group Theory for Randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985. (Cited on pages iv and vii.)

[BBGS14] Alp Bassa, Peter Beelen, Arnaldo Garcia, and Henning Stichtenoth. An Improvement of the Gilbert–Varshamov Bound Over Nonprime Fields. *IEEE Transactions on Information Theory*, 60(7):3859–3861, 2014. (Cited on page 77.)

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 14:1–14:17, 2018. (Cited on pages vi, viii, ix, 16, 18, 21, 22, 24, 25, 26, 27, 35, 36, 37, 38, 51, 67, 73, 74, 89, 95, 96, 109, 113 and 114.)

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018:46, 2018. (Cited on pages 28, 29, 34 and 75.)

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable Zero Knowledge with No Trusted Setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019. (Cited on pages vi, viii, 16, 27, 28, 29, 33, 34 and 53.)

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988. (Cited on page v.)

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying Program Executions Succinctly and in Zero knowledge. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013. (Cited on page vi.)

[BCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 287–304. IEEE Computer Society, 2015. (Cited on page vi.)

[BCG+17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive Oracle Proofs with Constant Rate and Query Complexity. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 40:1–40:15, 2017. (Cited on pages vi, viii, ix, 18, 27, 51, 53 and 81.)

[BCG+19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. Linear-Size Constant-Query IOPs for Delegating Computation. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 494–521. Springer, 2019. (Cited on pages vi, viii and 81.)

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-Time Arguments with Sublinear Verification from Tensor Codes. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA,*

November 16-19, 2020, Proceedings, Part II, volume 12551 of *Lecture Notes in Computer Science*, pages 19–46. Springer, 2020. (Cited on pages vi, vii, 20, 53, 74 and 114.)

[BCGT13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the Concrete Efficiency of Probabilistically-Checkable Proofs. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 585–594, New York, NY, USA, 2013. Association for Computing Machinery. (Cited on pages 17 and 53.)

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct Non-interactive Arguments via Linear Interactive Proofs. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 315–333. Springer, 2013. (Cited on page vi.)

[BCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed-Solomon Codes. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 900–909. IEEE, 2020. (Cited on pages viii, 23, 25, 27, 43, 67 and 113.)

[BCKL21] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Elliptic Curve Fast Fourier Transform (ECFFT) Part I: Fast Polynomial algorithms over all finite fields. *Electron. Colloquium Comput. Complex.*, page 103, 2021. (Cited on page viii.)

[BCL20] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. Zero-Knowledge Succinct Arguments with a Linear-Time Prover. *IACR Cryptol. ePrint Arch.*, page 1527, 2020. (Cited on pages vi, vii, 42, 74, 81, 113 and 114.)

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent Succinct Arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019. (Cited on pages vi, vii, viii, 6 and 27.)

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016. (Cited on pages vi, vii, viii, 5, 8, 9, 28 and 74.)

[BDN17] Amey Bhangale, Irit Dinur, and Inbal Livni Navon. Cube vs. Cube Low Degree Test. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 40:1–40:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (Cited on page 54.)

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-Deterministic Exponential Time Has Two-Prover Interactive Protocols. In *31st Annual Symposium on Foundations of*

Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I, pages 16–25. IEEE Computer Society, 1990. (Cited on pages vii and 52.)

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991. (Cited on pages vii, 4, 5, 42, 49, 52 and 54.)

[BGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, shorter PCPs and applications to coding. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 1–10. ACM, 2004. (Cited on pages 5, 17 and 18.)

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling Outside the Box Improves Soundness. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 5:1–5:32, 2020. (Cited on pages viii, 15, 25, 27, 34, 39, 43, 44, 45, 46, 67, 73 and 113.)

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131. ACM, 1988. (Cited on page vii.)

[BKK+13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 320–329. IEEE Computer Society, 2013. (Cited on pages ix and 81.)

[BKS18] Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-Case to Average Case Reductions for the Distance to a Code. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 24:1–24:23, 2018. (Cited on pages viii, 20, 25, 27, 49, 50, 67 and 113.)

[BN20] Sarah Bordage and Jade Nardi. Interactive Oracle Proofs of Proximity to Algebraic Geometry Codes. *Electron. Colloquium Comput. Complex.*, 27:165, 2020. (Cited on page 39.)

[BRS20] Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast Encoding of AG Codes over C_{ab} Curves, 2020. (Cited on page 78.)

[BS05] Alin Bostan and Eric Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, 2005. (Cited on page 112.)

[BS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006. (Cited on pages 51 and 53.)

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with Polylog Query Complexity. *SIAM J. Comput.*, 38(2):551–607, 2008. (Cited on pages viii, 14, 17, 18, 21, 25, 34, 52, 53, 68, 74 and 81.)

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-Efficient Low Degree Tests and Short PCPs via Epsilon-Biased Sets. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 612–621, New York, NY, USA, 2003. Association for Computing Machinery. (Cited on pages 53 and 54.)

[CMS17] Alessandro Chiesa, Peter Manohar, and Igor Shinkar. On Axis-Parallel Tests for Tensor Product Codes. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 39:1–39:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (Cited on pages 51 and 53.)

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct Arguments in the Quantum Random Oracle Model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 1–29. Springer, 2019. (Cited on pages vi and 28.)

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-Quantum Succinct Arguments. *Electron. Colloquium Comput. Complex.*, page 38, 2021. (Cited on page v.)

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and Transparent Recursive Proofs from Holography. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 769–793. Springer, 2020. (Cited on page viii.)

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. (Cited on pages viii, 17 and 18.)

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. (Cited on page 13.)

[DR04] Irit Dinur and Omer Reingold. Assignment Testers: Towards a Combinatorial Proof of the PCP-Theorem. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 155–164. IEEE Computer Society, 2004. (Cited on pages 5, 17 and 18.)

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *J. ACM*, 43(2):268–292, 1996. (Cited on page 52.)

[FHS94] Katalin Friedl, Zsolt Hátsági, and Alexander Shen. Low-Degree Tests. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, page 57–64, USA, 1994. Society for Industrial and Applied Mathematics. (Cited on page 52.)

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. (Cited on pages iv, v, 8 and 9.)

[Ful08] William Fulton. *Algebraic Curves: An Introduction to Algebraic Geometry*. Electronic edition, 2008. (Cited on page 82.)

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013. (Cited on page vi.)

[GGR09] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 13–22. ACM, 2009. (Cited on page 114.)

[GLR+91] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 32–42. ACM, 1991. (Cited on page 53.)

[GLS+21] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum SNARKs for R1CS. Cryptology ePrint Archive, Report 2021/1043, 2021. `https://ia.cr/2021/1043`. (Cited on pages vi and 114.)

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985. (Cited on pages iv, vii and 5.)

[Gol08] Oded Goldreich. *Computational complexity: A conceptual perspective*. Cambridge University Press, first edition, 2008. (Cited on pages 3 and 42.)

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. (Cited on pages 7 and 12.)

[Gop77] Valerii Denisovich Goppa. Codes associated with divisors. *Problemy Peredachi Informatsii*, 13(1):33–39, 1977. (Cited on pages 12 and 74.)

[GR11] Oded Goldreich and Dana Ron. On Proximity-Oblivious Testing. *SIAM Journal on Computing*, 40(2):534–566, 2011. (Cited on page 7.)

[Gro10] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010. (Cited on page vi.)

[Gro16] Jens Groth. On the Size of Pairing-Based Non-interactive Arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016. (Cited on page vi.)

[GS86] Shafi Goldwasser and Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68. ACM, 1986. (Cited on page iv.)

[GS95] Arnaldo Garcia and Henning Stichtenoth. A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vladut bound. *Inventiones Mathematicae*, 121(1):211–222, 1995. (Cited on page ix.)

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inf. Theory*, 45(6):1757–1767, 1999. (Cited on page 23.)

[GS02] Oded Goldreich and Madhu Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 13–22. IEEE Computer Society, 2002. (Cited on page 54.)

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 99–108. ACM, 2011. (Cited on pages v and 9.)

[HY18] Chuangqiang Hu and Shudi Yang. Multi-point codes over Kummer extensions. *Designs, Codes and Cryptography*, 86:211–230, 2018. (Cited on page 100.)

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient Arguments without Short PCPs. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 278–291. IEEE Computer Society, 2007. (Cited on page vi.)

[IMS12] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge pcps. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2012. (Cited on page v.)

[JPRZ04] Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing Low-Degree Polynomials over Prime Fields. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 423–432. IEEE Computer Society, 2004. (Cited on page 54.)

[Kan86] Ernst Kani. The Galois-module structure of the space of holomorphic differentials of a curve. *Journal für die reine und angewandte Mathematik*, 367:187–206, 1986. (Cited on pages 79 and 90.)

[Kil92] Joe Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992. (Cited on pages v, vi and 8.)

[KPV19] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://ia.cr/2019/1400`. (Cited on pages vi and viii.)

[KR04] Tali Kaufman and Dana Ron. Testing Polynomials over General Fields. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 413–422. IEEE Computer Society, 2004. (Cited on page 54.)

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008. (Cited on pages vi and 5.)

[Lac87] Gilles Lachaud. Sommes d'Eisenstein et nombre de points de certaines courbes algébriques sur les corps finis. *C. R. Acad. Sci. Paris*, 305, 01 1987. (Cited on page 98.)

[Lac92] Gilles Lachaud. Artin-Schreier curves, exponential sums, and coding theory. *Theoretical Computer Science*, 94(2):295–310, 1992. (Cited on page 77.)

[LANHC16] Sian-Jheng Lin, Tareq Y. Al-Naffouri, Yunghsiang S. Han, and Wei-Ho Chung. Novel Polynomial Basis With Fast Fourier Transform and Its Application to Reed–Solomon Erasure Codes. *IEEE Transactions on Information Theory*, 62(11):6284–6299, 2016. (Cited on page 28.)

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10. IEEE Computer Society, 1990. (Cited on pages iv, vii and 81.)

[Lip13] Helger Lipmaa. Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors,

Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I, volume 8269 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2013. (Cited on page vi.)

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, 1997. (Cited on page 22.)

[Mah04] Hiren Maharaj. Code Construction on Fiber Products of Kummer Covers. *Information Theory, IEEE Transactions on*, 50:2169 – 2173, 10 2004. (Cited on page 98.)

[Mei13] Or Meir. IP = PSPACE Using Error-Correcting Codes. *SIAM J. Comput.*, 42(1):380–403, 2013. (Cited on pages 10 and 74.)

[Mic95] Silvio Micali. Computationally-Sound Proofs. In Johann A. Makowsky and Elena V. Ravve, editors, *Proceedings of the Annual European Summer Meeting of the Association of Symbolic Logic, Logic Colloquium 1995, Haifa, Israel, August 9-18, 1995*, volume 11 of *Lecture Notes in Logic*, pages 214–268. Springer, 1995. (Cited on pages v, vi and 8.)

[Mic00] Silvio Micali. Computationally Sound Proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000. (Cited on pages vi and 8.)

[Mie09] Thilo Mie. Short PCPPs Verifiable in Polylogarithmic Time with O(1) Queries. *Annals of Mathematics and Artificial Intelligence*, 56(3–4):313–338, August 2009. (Cited on pages vi, 17, 51, 52, 53 and 81.)

[Mor91] Carlos Moreno. *Algebraic Curves over Finite Fields*. Cambridge Tracts in Mathematics. Cambridge University Press, 1991. (Cited on page 82.)

[Mos10] Dana Moshkovitz. An Alternative Proof of The Schwartz-Zippel Lemma. *Electron. Colloquium Comput. Complex.*, page 96, 2010. (Cited on page 13.)

[MP93] Carlos Munuera and Ruud Pellikaan. Equality of geometric Goppa codes and equivalence of divisors. *Journal of Pure and Applied Algebra*, 90(3):229 – 252, 1993. (Cited on page 95.)

[MQS15] Ariane M. Masuda, Luciane Quoos, and Alonso Sepúlveda. One- and Two-Point Codes over Kummer Extensions. *arXiv e-prints*, page arXiv:1510.06425, October 2015. (Cited on page 98.)

[MR08] Dana Moshkovitz and Ran Raz. Sub-Constant Error Low Degree Test of Almost-Linear Size. *SIAM J. Comput.*, 38(1):140–180, 2008. (Cited on pages 53 and 54.)

[MS77] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-correcting Codes*. Mathematical Library. North-Holland Publishing Company, 1977. (Cited on pages 9 and 10.)

[MX13] Mohammad Mahmoody and David Xiao. Languages with efficient zero-knowledge pcps are in SZK. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2013. (Cited on page v.)

[NN90] Joseph Naor and Moni Naor. Small-bias Probability Spaces: Efficient Constructions and Applications. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 213–223. ACM, 1990. (Cited on page 42.)

[Pan94] Victor Y. Pan. Simple Multivariate Polynomial Multiplication. *J. Symb. Comput.*, 18(3):183–186, 1994. (Cited on page 55.)

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252. IEEE Computer Society, 2013. (Cited on page vi.)

[PS94] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 194–203. ACM, 1994. (Cited on pages viii, 17, 25 and 52.)

[PSW91] Ruud Pellikaan, Ba-zhong Shen, and Gerhard J. M. van Wee. Which linear codes are algebraic-geometric? *IEEE Transactions on Information Theory*, 37(3):583–602, 1991. (Cited on page 101.)

[RR20] Noga Ron-Zewi and Ron D. Rothblum. Local Proofs Approaching the Witness Length [Extended Abstract]. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 846–857. IEEE, 2020. (Cited on pages vi, viii, 27, 52, 53, 74, 81 and 114.)

[RR21] Noga Ron-Zewi and Ron Rothblum. Proving as Fast as Computing: Succinct Arguments with Constant Prover overhead. *Electron. Colloquium Comput. Complex.*, page 180, 2021. `https://eccc.weizmann.ac.il/report/2021/180`. (Cited on pages vi and 114.)

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62. ACM, 2016. (Cited on pages vi, vii and 5.)

[RS60] Irving S. Reed and Gustave Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. (Cited on page 11.)

[RS92] Ronitt Rubinfeld and Madhu Sudan. Self-Testing Polynomial Functions Efficiently and Over Rational Domains. In Greg N. Frederickson, editor, *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27-29 January 1992, Orlando, Florida, USA*, pages 23–32. ACM/SIAM, 1992. (Cited on pages 53 and 54.)

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust Characterizations of Polynomials with Applications to Program Testing. *SIAM J. Comput.*, 25(2):252–271, 1996. (Cited on pages 53 and 54.)

[RS97] Ran Raz and Shmuel Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 475–484. ACM, 1997. (Cited on pages 52, 53 and 54.)

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. (Cited on pages 14 and 33.)

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980. (Cited on page 13.)

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. (Cited on page iv.)

[Sta] StarkDEX. `https://www.starkdex.io/`. (Cited on pages viii and 27.)

[Sta21a] StarkWare. ethSTARK Documentation. Cryptology ePrint Archive, Report 2021/582, 2021. `https://ia.cr/2021/582`. (Cited on pages vi, 29, 34 and 75.)

[Sta21b] StarkWare. ethSTARK Documentation. *IACR Cryptol. ePrint Arch.*, page 582, 2021. (Cited on page viii.)

[Sti09] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Graduate Texts in Mathematics. Springer, second edition, 2009. (Cited on pages 77, 82, 86, 101 and 102.)

[Tha22] Justin Thaler. *Proofs, Arguments, and Zero-Knowledge*. 2022. Available at `https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf`, cited version is dated 31 March, 2022. (Cited on page 3.)

[TT14] Saeed Tafazolian and Fernando Torres. On the curve $y^n = x^m + x$ over finite fields. *Journal of Number Theory*, 145:51–66, 2014. (Cited on page 111.)

[TVN07] Michael Tsfasman, Serge Vladut, and Dmitry Nogin. *Algebraic Geometric Codes: Basic Notions*. American Mathematical Society, USA, 2007. (Cited on page 82.)

[TVZ82] M. A. Tsfasman, S. G. Vlăduţ, and Th. Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Math. Nachr.*, 109:21–28, 1982. (Cited on pages 12, 74 and 77.)

[Val08] Paul Valiant. Incrementally Verifiable Computation or Proofs of Knowledge Imply time/space efficiency. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008. (Cited on pages v, vi and 8.)

[Vid15] Michael Viderman. A combination of testability and decodability by tensor products. *Random Struct. Algorithms*, 46(3):572–598, 2015. (Cited on pages 12, 51 and 53.)

[VSBW13] Victor Vu, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. A Hybrid Architecture for Interactive Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 223–237. IEEE Computer Society, 2013. (Cited on page 55.)

[Zca] Zcash. `https://z.cash`. (Cited on page vi.)

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. (Cited on page 13.)

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 859–876. IEEE, 2020. (Cited on pages vi and viii.)

ECOLE
DOCTORALE

Titre: Protocoles efficaces pour tester la proximité à des codes algébriques

Mots clés: Preuves vérifiables de manière probabiliste, preuves interactives, tests de bas degré, tests de proximité, codes correcteurs d'erreurs, géométrie algébrique

Résumé: Les preuves vérifiables de manière probabiliste (PCP, de l'anglais *probabilistically checkable proofs*), les preuves interactives (IP, pour *interactive proofs*) ou encore les preuves à divulgation nulle de connaissance (*zero-knowledge proofs*) ont la particularité d'admettre une vérification probabilististe. Ces systèmes de preuves probabilistes interviennent dans les constructions de schémas de calcul vérifiable, des protocoles cryptographiques permettant de vérifier très rapidement qu'un long calcul a été correctement effectué. En 2016, un nouveau modèle de preuve a été introduit par Ben-Sasson, Chiesa et Spooner: celui des preuves interactives par oracle (IOP, pour *interactive oracle proofs*). Ce modèle généralise à la fois les PCPs et les IPs et a suscité beaucoup d'intérêt depuis son introduction. Le modèle IOP a mené à d'intéressants résultats théoriques sur les arguments non-interactifs succincts et transparents ainsi qu'à des déploiements industriels.
Un problème récurrent dans les constructions de systèmes de preuves probabilistes est celui de tester efficacement la proximité à un code correcteur d'erreurs. Le but est de déterminer si un certain mot appartient à un code linéaire donné, ou bien s'il est éloigné de tout mot de ce code. Les tests de proximité à des codes polynomiaux peuvent être interprétés comme des tests de bas degré. Par exemple, un important sous-protocole utilisé dans de nombreuses constructions pratiques est un *IOP of Proximity* pour les codes de Reed-Solomon (Ben-Sasson *et al.*, ICALP 2018).
Dans cette thèse, nous proposons dans le modèle IOP des protocoles permettant de vérifier la proximité à des codes correcteur d'erreurs. En nous inspirant du test de proximité pour les codes de Reed-Solomon de Ben-Sasson *et al.*, nous commençons par formuler un cadre abstrait et générique pour construire des *IOPs of Proximity* pour des codes linéaires et en analysons formellement les propriétés. Nous appliquons ensuite cette méthodologie à différentes familles de codes généralisant les codes de Reed-Solomon. Il s'agit d'une part de codes définis à partir d'évaluations de polynômes multivariés et, d'autre part, de codes de géométrie algrébrique définis sur des courbes. Nos protocoles permettent de tester la proximité à des codes présentant des propriétés attrayantes par rapport aux codes de Reed-Solomon (telles que des alphabets de petite taille), tout en ayant une efficacité similaire à la construction de Ben-Sasson *et al*.

Title: Efficient protocols for testing proximity to algebraic codes

Keywords: Interactive proof systems, probabilistic proof systems, low degree tests, proximity testing, error-correcting codes, algebraic geometry

Abstract: Probabilistic proof systems, such as probabilistically checkable proofs, interactive proofs, and zero-knowledge proofs, feature the common characteristic of having a probabilistic verification procedure. Notably, such proof systems are at the heart of cryptographic protocols that enable polylogarithmic-time verification of very long computations. Generalizing both PCPs and IPs, the Interactive Oracle Proof (IOP) model has been introduced in 2016 by Ben-Sasson, Chiesa and Spooner. The IOP model has attracted a lot of interest since its introduction, leading to both interesting theoretical results related to efficient transparent succinct non-interactive arguments and industrial deployments.
A recurrent problem in constructions of probabilistic proof systems is that of testing proximity to an error-correcting code. The goal is to determine whether a certain word belongs to a given linear code, or if it is far from any codeword of that code. Proximity tests for polynomial codes are often called low degree tests. A notable building-block of several IOP-based constructions is a concretely efficient IOP of Proximity for testing proximity to Reed-Solomon codes (Ben-Sasson *et al.*, ICALP 2018).
In this thesis, we propose protocols in the IOP model for verifying proximity to error-correcting codes. Based on the proximity test for Reed-Solomon codes designed by Ben-Sasson *et al.*, we formulate and analyze an abstract and generic framework to construct IOPs of Proximity for linear codes. We then apply this methodology to different families of codes that generalize Reed-Solomon codes. These are, on the one hand, codes defined from evaluations of multivariate polynomials and, on the other hand, algebraic geometry codes defined on curves. Our protocols have similar efficiency parameters compared to the construction of Ben-Sasson *et al.*, and allow efficient proximity testing for families of codes with attractive properties compared to Reed-Solomon codes (such as small alphabet sizes).

Institut Polytechnique de Paris
91120 Palaiseau, France