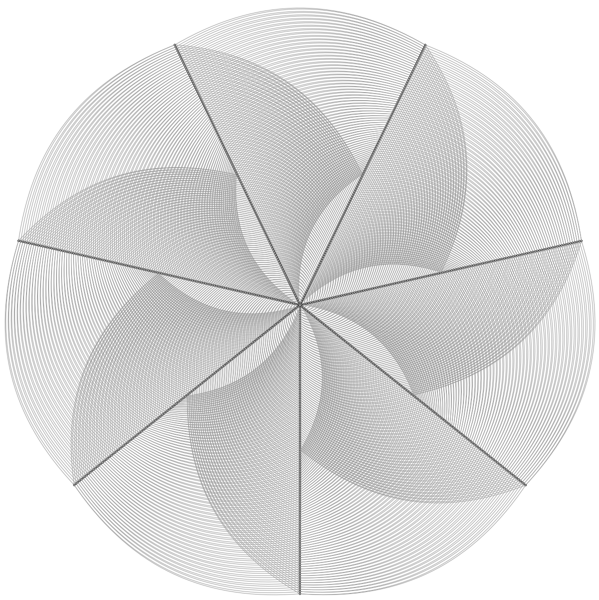
Internship report

Short PCP-like proof simultaneous verification using interleaved Reed-Solomon codes

Hugo Delavenne

21 March — 19 August 2022

Under the supervision of
Daniel Augot



Contents

Part I

Introduction

1 STARK protocols

When a server interacts with a client, it cannot always be trusted and the client may want to have proofs that the server is not lying and is honnestly executing its task. The issue is that the client is computationally weak, so it can't repeat the same computation to check the output. Likewise, the server doesn't want to perform too much extra computations to be able to prove that it is honnest. This happens for example in blockchain where the proof of verification of the interactions must be added to blockchain.

$\mathbf{NP}$ is a first model of efficient verification in which we consider that a prover gives a proof to be checked deterministically in polnomial time. Probabilistic check is introduced with $\mathbf{PCP}$, with very impressive theoretical result [BFLS91]. The idea is that the verifier doesn't read the whole proof to decide it is it accepted or not. This theorem was completely unpractical until [BS08] which created $\mathbf{PCP}$ proofs of quasilinear length. A new model adding interaction with the prover, $\mathbf{IOP}$, is introduced in [BCS16], leading to the creation of practical efficient protocols [BBHR18a, BBHR18b, BKS18, BGKS20, BCI+20]. Those protocols are called STARKs. It stands for Scalable Transparent ARgument of Knowledge. An argument of knowledge is a proof that the prover correctly executed a computation, scalable means that the proof is logarithmic in the size of the computation, and transparent means that the protocol does not rely on cryptographic asumptions.

1.1 Scientific context

Formally, $\mathbf{PCP}$ restricts the verifier to draw a bounded number of bits and to read a limited number of bits of the proof. The $\mathbf{PCP}$ theorem states that the verifier only require to read a constant number of bits of the proof to check it, independently of the size of the problem.

Definition 1.1 PCP (Probabilistically Checkable Proof)

A language $\mathcal{L} \subseteq \Sigma^*$ is in $\mathbf{PCP}[r(n), q(n)]$ if there exists a polynomial time verifier V, using at most $r(n)$ internal randomness bits and reading only $q(n)$ bits of its input proof, such that

- Perfect completeness: $\forall x \in \mathcal{L}, \exists \pi \text{ proof}, \mathbb{P}\big(V(x,\pi) = \text{accept}\big) = 1$
- Soundness: there exists $s < 1/2$ such that $\forall x \notin \mathcal{L}, \forall \pi \text{ proof}, \mathbb{P}\big(V(x,\pi) = \text{accept}\big) < s$

where the probability is taken on the internal randomness of the verifier.

Theorem 1.2 PCP theorem [BFLS91]

$\mathbf{PCP}[O(\log n), O(1)] = \mathbf{NP}$
$\mathbf{PCP}[O(\text{poly}(n)), O(1)] = \mathbf{NEXP}$

The issue is that the proof itself is huge and requires a lot of computation to be generated, much more than finding a general proof, so this result itself is only theoretical. In [BS08], Ben-Sasson and Sudan managed to generate $\mathbf{PCP}$ proofs of quasilinear length and that can be verified by reading a polylog number of bits. This result opened the door to creating practical $\mathbf{PCP}$-like theorems. Another approach for proof systems is $\mathbf{IP}$.

Definition 1.3 IP (Interactive Proofs)

A language $\mathcal{L} \subseteq \Sigma^*$ is in $\mathbf{IP}$ if there exists a polynomial time verifier V having one or more interactions with an unboundedly powerful prover such that

- Perfect completeness: $\forall x \in \mathcal{L}, \exists P \text{ prover}, \mathbb{P}(V(x, P(x)) = \text{accept}) = 1$

- Soundness: there exists $s < 1/2$ such that $\forall x \notin \mathcal{L}, \forall P, \mathbb{P}(V(x, P(x)) = \text{accept}) < s$

where the probability is taken over the internal randomness of the verifier.

For cryptographic assumptions, we will require in the following that the soundness is at most 2^{-128} or 2^{-256}.

Theorem 1.4 $\mathbf{IP} = \mathbf{PSPACE}$ [LFKN92]

$\mathbf{IP} = \mathbf{PSPACE}$

A more recent approach [BCS16] is the fusion of the properties of $\mathbf{PCP}$ and $\mathbf{IP}$, where the verifier interacts with a prover, and the prover gives oracle access to its proofs, so that the verifier does not read them entirely. Having such oracle access makes this new class bigger than $\mathbf{IP}$.

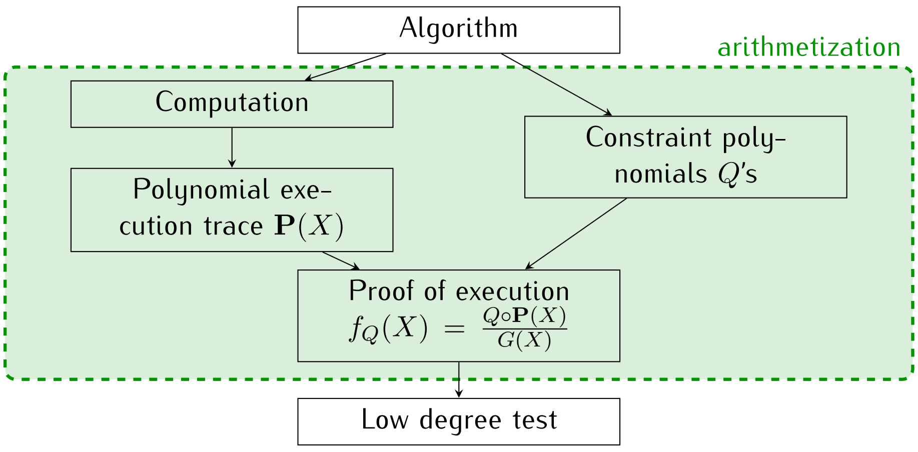
Theorem 1.5 $\mathbf{IOP} = \mathbf{NEXP}$ [BCS16]

$\mathbf{IOP} = \mathbf{NEXP}$

Since the verifier's randomness is public-coin, it is possible to use a Fiat-Shamir heuristic to turn the interactive proof into a non-interactive one.

1.2 Arithmetization

The summary of the construction is the following.


Algorithm
arithmetization
Computation
Polynomial execution trace $\mathbf{P}(X)$
Constraint polynomials Q's
Proof of execution $f_Q(X) = \frac{Q \circ \mathbf{P}(X)}{G(X)}$
Low degree test

The prover constructs the whole arithmetization and the verifier only constructs the constraint polynomials in order to be able to perform consistency checks with the proof of execution.

The prover executes a computation in T steps, using R registers (memory cells). It has an execution trace which is R functions $r_1, ..., r_R$ from H to $\mathbb{F}$, where $|H| = T + 1$ is detailled below. We encode the transitions in such a way that it is possible to be convinced of the validity of the computation by checking only a few values. In other words we encode the execution trace so that if there are some errors then the encoding has lots of errors.

The computation trace is valid if it respects the constraints induced by the asked computation. There are two kind of constraints on the registers:

- The local constraints tell what value must have some registers at some specific time of the execution, typically the initial and final values.
- The constraint polynomials tell what transitions of register states are allowed. They are of the form $Q(X_1, ..., X_R, Y_1, ..., Y_R)$ where X_i represents the value of register i at time t, and Y_i represents the value of the same register i at time $t+1$. These polynomials must have as only roots the valid values of transitions.

Example 1.6 Constraints for the computation of square Fibonacci

Consider the square Fibonacci sequence where $f_0 = f_1 = 1$ and $f_{i+2} = f_{i+1}^2 + f_i^2 \mod 96769$. A verifier wants to know the 10^9th term of the sequence, so it asks a prover that claims that the result is 36452. The prover works in $\mathbb{F} = \mathbb{Z}/96769\mathbb{Z}$ and only needs one register for the f_i's and one for the "f_{i-1}'s", that we call g_i and that is such that $g_{i+1} = f_i$. The constraints will be here the following.

- The local constraints are $f_0 = 1$, $f_1 = 1$ and $f_{10^9} = 36452$.
- The transition constraints are $f_{i+1} - f_i^2 - g_i^2 = 0$ and $g_{i+1} - f_i = 0$. So the constraint polynomials are

$$Q_1(X_1, X_2, Y_1, Y_2) = Y_1 - X_1^2 - X_2^2 \qquad Q_2(X_1, X_2, Y_1, Y_2) = Y_2 - X_1.$$

Once the prover has the constraint polynomials and the execution trace, it creates the proof polynomials.

For arithmetic purposes, we set H to be a multiplicative subgroup of $\mathbb{F}^*$ of size $T+1$ with generator g. With this, we go from time t to time $t+1$ by multiplying by g.

The prover computes an interpolation of the execution traces as polynomials P_i's: $\forall t, P_i(g^t) := r_i(g^t)$. For each constraint polynomial $Q(X_1, ..., X_R, Y_1, ..., Y_R)$, the prover creates the associated univariate composed polynomial

$$(Q \circ \mathbf{P})(X) := Q(\mathbf{P}(X), \mathbf{P}(gX)),$$

with $\mathbf{P}(X) := (P_1(X), ..., P_R(X)) \in (\mathbb{F}[X])^R$.

With this construction, if the computation is valid then the composed polynomials have all the g^t as roots, so $G(X) := \prod_{1 \leqslant t \leqslant T}(X - g^t)$ divides the composed constraint polynomials, i.e.

$$\frac{(Q \circ \mathbf{P})(X)}{G(X)} \in \mathbb{F}(X)$$

is a polynomial iff the product divides the composed polynomial. We will call this a proof polynomial. Since $G(X) = X^{T+1} - 1$, it allows the verifier to efficiently compute this product to perform consistency checks. Now we have a set of proof polynomials, and if one wants only one proof polynomial, one can take a random linear combination of these polynomials.

For a valid computation, if d is the maximal degree of the constraint polynomials then the degree of a composed polynomial is at most dT and the degree of a proof polynomial is at most $dT - T$. In fact, seen as a function in a finite field, a rational fraction can always be seen as a polynomial. If the computation is not valid, then we can show that the proof polynomial isn't any function. This is a gap promise that we will use: the proof polynomial is either low-degree or very different from a low-degree polynomial (see Section 1.3.3).

1.3 Low degree testing

1.3.1 Reed-Solomon codes

Linear error-correcting codes are vector spaces over $\mathbb{F}^n$ in which we consider some parameters like the ratio of the number of elements of the code over the number of elements of the ambiant space (the rate of the code), or the minimal distance between two elements for some distance. Reed-Solomon codes are linear codes defined as evaluation of low-degree polynomials on a given domain.

Definition 1.7 **Reed-Solomon code**

The code RS$[\mathcal{D}, \rho]$ is $\{f : \mathcal{D} \to \mathbb{F} \mid \exists P(X) \in \mathbb{F}[X]_{<\rho|\mathcal{D}|}, P_{|\mathcal{D}} = f\}$, with $\mathcal{D} \subseteq \mathbb{F}$ the evaluation domain and $\rho \in [0, 1]$ the rate of the code.

Reed-Solomon codewords can be considered as functions interpolating polynomials, or as vectors over $\mathbb{F}^{|\mathcal{D}|}$. The distance used here is the Hamming distance.

Definition 1.8 **Hamming distance**

Let $u, v \in \mathbb{F}^n$. The Hamming distance between u and v, denoted $\Delta(u, v)$ is the number of indexes $i \in [\![1, n]\!]$ such that $u_i \neq v_i$, divided by n so that $\Delta(u, v) \in [0, 1]$.

The Hamming distance between $u \in \mathbb{F}^N$ and $S \subseteq \mathbb{F}^n$ is $\Delta(u, S) := \min_{v \in S} \Delta(u, v)$.

The minimal distance between any two elements of a set S is denoted $\Delta(S) := \min_{u \neq u' \in S} \Delta(u, u')$.

1.3.2 FRI protocol [BBHR18a]

FRI is the acronym for Fast Reed-Solomon IOPP (IOPP being an acronym for IOP of Proximity). It is an IOP protocol for convincing a verifier that a given function is a Reed-Solomon codeword, by using techniques inspired from FFT [BBHR18a].

To prove that a function f is a polynomial of degree at most k it suffices, for a divide and conquer approach, to prove that the even part f_0 and the odd part f_1 of f are polynomials of degree at most $k/2$. Even and odd parts means that $f(X) = f_0(X^2) + X f_1(X^2)$, i.e. $f_0(Y) = \frac{f(X)+f(-X)}{2}$ and $f_1(Y) = \frac{f(X)-f(-X)}{2X}$, for $Y = X^2$.

However, testing the degrees of both f_0 and f_1 costs as much as testing the degree of f, so the idea is to test the degree of $f_0 + \alpha f_1$ for a random $\alpha \in \mathbb{F}$. If f_0 and f_1 are indeed polynomials of degree $< k/2$ then so is $f_0 + \alpha f_1$ with probability 1, and if f_0 or f_1 is a polynomial of high degree then $f_0 + \alpha f_1$ is of high degree with high probability over the choice of α.

This linear combination of the even and odd parts of f over α is called the folding of f.

Notation 1.9 **Folding**

For $f : \mathbb{F}^* \to \mathbb{F}$, with $f_0 : y \mapsto \frac{f(x)+f(-x)}{2}$ and $f_1 : y \mapsto \frac{f(x)-f(-x)}{2x}$, where $x^2 = y$, denote the folding of f over $\alpha \in \mathbb{F}$ by $\mathbf{Fold}[f, \alpha] := f_0 + \alpha f_1$.

Testing the degree of a polynomial over a domain $\mathcal{D}$ reduces to testing the degree over a domain of size $|\mathcal{D}|/2$. Therefore we need the domains to be the 2^ith roots of the unit, so that the folding is defined over the 2^{i-1}th roots of the unit.

Definition 1.10 **FRI protocol**

Let $r \leqslant n \in \mathbb{N}$. Let $\mathcal{D}_i \subseteq \mathbb{F}$ be the set of 2^{n-i}th roots of the unit, so that $\mathcal{D}_{i+1} = \{x^2 \mid x \in \mathcal{D}_i\}$.
Define the code V_i as $\mathsf{RS}[\mathcal{D}_i, \rho]$. Let $\rho = 2^{r-n}$ so that $2^r = \rho|\mathcal{D}_0|$.
Prover wants to prove that a given function $f_0 : \mathcal{D}_0 \to \mathbb{F}$ is in $\mathsf{RS}[\mathcal{D}_0, \rho]$. There are two phases during the FRI protocol.

Commit phase. The verifier sends a random $\alpha \in \mathbb{F}$ and the prover commits $\mathbf{Fold}[f, \alpha]$. This process is repeated until the degree of the polynomial is zero.

$$\begin{tikzcd}
 & \text{Prover} & & \text{Verifier} \\
\text{commit} & \begin{array}{c} f_1 : \mathcal{D}_1 \to \mathbb{F} \\ (f_1 = \mathbf{Fold}[f_0, \alpha_0]) \end{array} \arrow[rr, shift right, "f_1"'] & & \alpha_0 \xleftarrow{\$} \mathbb{F} \arrow[ll, shift right, "\alpha_0"'] \\
 & \vdots & \vdots & \vdots \\
 & \begin{array}{c} f_r : \mathcal{D}_r \to \mathbb{F} \\ (f_r = \mathbf{Fold}[f_{r-1}, \alpha_{r-1}]) \end{array} \arrow[rr, shift right, "f_r"'] & & \alpha_{r-1} \xleftarrow{\$} \mathbb{F} \arrow[ll, shift right, "\alpha_{r-1}"']
\end{tikzcd}$$

Query phase. The verifier checks if each commited polynomial is indeed the folding of the original function. This process is applied successively at each level of folding by taking , and is repeated several times to increase the soundness.

query

$$\left.\begin{array}{c}\text{Verifier chooses } s_0 \overset{\$}{\leftarrow} \mathcal{D}_0\text{, defines } s_{i+1} := s_i^2 \text{ and accepts if} \\ f_1(s_1) = \textbf{Fold}[f_0, \alpha_0](s_1) \\ \vdots \\ f_r(s_r) = \textbf{Fold}[f_{r-1}, \alpha_{r-1}](s_r).\end{array}\right)\text{repeat}$$

This protocol requires a logarithmic number of random bits from the verifier, a logarithmic number of rounds of communication, and the verifier only looks at three elements of the proof to check the consistency of f_{i+1} with f_i on one value.

The hard part, left to Sections 2.2 and 3.1, is now to compute a bound on the soundness.

1.3.3 Gap promise

The above arithmetization is such that if the computation respects the constraints then the proof polynomial is low-degree (in the code), but if the computation does not respect the constraints then the proof polynomial is far from the code.

Lemma 1.11 Gap promise

Let $\mathcal{D} \subseteq \mathbb{F}$, $\rho \in [0, 1]$ and $C = \mathsf{RS}[\mathcal{D}, \rho]$. Let $P_1(X), P_2(X) \in \mathbb{F}[X]$. Suppose that $\forall x \in \mathcal{D}, P_2(x) \neq 0$ and

$$P_2(X) \text{ does not divide } P_1(X). \tag{1}$$

Then with $f(x) := P_1(x)/P_2(x)$ for $x \in \mathcal{D}$, we have $\Delta(f, C) \geqslant 1 - \max\left(\frac{\deg P_1}{|\mathcal{D}|}, \rho + \frac{\deg P_2}{|\mathcal{D}|}\right)$.

Proof.
Let $\delta := \Delta(f, C)$ and $g \in C$ such that $\Delta(f, g) = \delta$. Let $I = \{x \in \mathcal{D} \mid f(x) = g(x)\}$. Then $|I| = (1 - \delta)|\mathcal{D}|$ and for all $x \in I$, $P_1(x) - P_2(x)g(x) = 0$. By Equation (1) we know that $P_1(X) - P_2(X)g(X)$ is not null, so $|I| \leqslant \deg(P_1(X) - P_2(X)g(X)) \leqslant \max(\deg P_1(X), \rho|\mathcal{D}| + \deg P_2(X))$. Thus the result.

$P_1(X)$ represents the composed polynomial $(Q \circ \mathbf{P})(X)$ and $P_2(X)$ represents the product $G(X)$. Equation (1) means exactly that there is a constraint that is not satisfied at some time t.

In the arithmetization we have that the composed polynomial has degree at most $T \deg Q$, the product has degree T and $\rho = \frac{T}{|\mathcal{D}|}$, so the gap promise is

$$\delta \geqslant 1 - \frac{T}{|\mathcal{D}|} \max(2, \deg Q).$$

2 Objective of the internship

Other arithmetizations can use different error-correcting codes to achieve different properties, like algebraic geometry codes [BLNR22]. The objective of the internship was to create a protocol similar to FRI for Interleaved Reed-Solomon codes.

It is motivated by the fact that in practice, several proofs must be done for several codewords, so maybe them can help to improve parameters as it does in coding theory.

2.1 Interleaved codes

In the practical use of error-correcting codes, it often happens to encode successively several messages. And in some models of error, we consider that an error occurs on successive bits. So an idea to improve the decodability is to interleave the messages in order to have only a bit of each message with one error (which we can correct), instead of having one message fully erroneous.

Definition 2.1 Interleaved error-correcting code

Given a code $C \subseteq \mathbb{F}^n$, its ℓ-interleaved code associated is

$$\left\{ \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{\ell,1} & u_{\ell,2} & \cdots & u_{\ell,n} \end{pmatrix} \middle| u_1, \dots, u_\ell \in C \right\}.$$

IRS$[\mathcal{D}, \rho, \ell]$ denotes the ℓ-interleaved code on the code RS$[\mathcal{D}, \rho]$.

The difference between the interleaved code and a simple concatenation of codewords is the distance. The distance between two matricial words u and v is the number of indices i such that the ith column of u is different from the ith column of v. Equivalently the code can be considered on the alphabet $\mathbb{F}^\ell$ in order to use the same Hamming distance.

2.2 Maximum distance and list-decoding

Denote by $\delta \in [0, 1]$ the gap promise. The soundness of the protocol is a function increasing in δ. However the proofs of soundness are valid only for bounded values of δ. RS$[\mathcal{D}, \rho]$ has a property telling that a word from the ambiant space is at distance at most $1 - \rho$ from any codeword (by interpolation), so $1 - \rho$ is the largest theoretical bound. But all the formulas for soundness that have been found are only proven for δ smaller. The best results, in [BCI+20], achieves $\delta \leqslant \delta_{\max} = 1 - \sqrt{\rho}$. This limitation is due to a code-related technique that is used in the proofs of the soundness, which is called list-decoding.

Usually, when using error-correcting codes, one wants to be able to recover the only correct codeword associated to an erroneous received word. This is only possible if the error introduced is small, otherwise there can be several choices of codewords to correct the received word. Getting a list of the closests codewords to a received word is called list-decoding that word.

Definition 2.2 List-decodability

Let Σ be a finite alphabet and $S \subseteq \Sigma^n$. S is said to be (δ, μ)-list-decodable if any ball of Σ^n of radius δ (for the Hamming distance) contains at most μ elements of S, i.e.

$$\forall u \in \Sigma^n, |B(u, \delta) \cap S| \leqslant \mu.$$

In the proofs of soundness we have to reconstruct codewords from the received proof polynomial, so the proofs use list-decoding, and they are therefore limited to the range of nice list-decoding. An important and very general result for list-decoding is the Johnson bound, which is a general bound on the size of the list-decoding for general codes (not even linear).

Theorem 2.3 Johnson bound

For $\varepsilon \in]0, 1]$, let $J_\varepsilon : \delta \mapsto 1 - \sqrt{1 - \delta(1 - \varepsilon)}$. If Σ is a finite alphabet, $n \in \mathbb{N}$, $S \subseteq \Sigma^n$, and $\varepsilon \in]0, 1]$, then S is $(J_\varepsilon(\Delta(S)), \frac{1}{\varepsilon})$-list-decodable.

This result is used as such in the proofs of soundness of [BKS18] and [BGKS20]. It bounds δ to be smaller than $\Delta(S)$, and the soundness is a decreasing function of the size of the list. Interleaved codes may be interesting to achieve a better bound.

2.3 Interest of interleaved Reed-Solomon codes

An important property of IRS$[\mathcal{D}, \rho, \ell]$ is that an error on at most $\frac{\ell}{\ell+1}(1 - \rho)$ columns and that is uniformly distributed on each coordinate can be uniquely corrected with high probability.

Proposition 2.4 Probabilistic decoding of an IRS [Zap20]

There is an algorithm such that, given an erroneous word received $u = v + e$, with $v \in$ IRS, if e has at most $\frac{\ell}{\ell+1}(1 - \rho)$ non-null columns and if each non-null column of e is uniformly distributed, the algorithm to recover v from u with probability $O(1/|\mathbb{F}|)$, probability taken on e.

This property looks like some "probabilistic list-decoding", telling that given a received word u, with some probability $p = O(1/|\mathbb{F}|)$, the ball of radius $\delta = \frac{\ell}{\ell+1}(1-\rho)$ contains at most $\mu = 1$ codewords. The field $\mathbb{F}$ will be chosen to be really huge in practice, so p is somehow negligible. The maximum distance δ is far better than any deterministic known result. And $\mu = 1$ is also far better any $1/\varepsilon$ that we would get with the Johnson bound. The first part of the internship was to revisit the different proofs to replace the use of the Johnson bound by a hypothesis of probabilistic list-decoding.

The issue with Proposition 2.4 is that the probability is somehow taken on the random choice of u, i.e. in our context on the proof polynomial. Since the prover is adversarial, it could take u's that does not satisfy this property.

Objective of the internship

Generalize proofs of soundness of the FRI protocol so that they can be used to build a similar protocol with interleaved codes, with a better bound on the maximum distance $\delta_{\max}$.

This objective has been achieved in Part II, modulo finding a suitable revisiting of Proposition 2.4. Therefore, as advised by Daniel Augot, I worked on the last part of my internship, in Part III, on understanding the use of De Bruijn graphs in some arithmetization compared to the other arithmetization.

Part II
FRI protocol with IRS

3 FRI protocol

The protocol described in Section 1.3.2 respects perfect completeness by construction, and for a practical use we want the soundness to be cryptographically small. To achieve this we need to compute bounds on the soundness, which will be a function of the gap promise δ and of the parameters (the field, the number of repetition of the query phase, and other parameters like ε that will appear next).

3.1 Soundness of the FRI protocol [BKS18]

In this section we use the notations introduced in Definition 1.10.

During the commit phase, the bad thing that can happen is that f is far from the code, yet the random value α drawn by the verifier to fold f makes $\mathbf{Fold}[f, \alpha]$ closer to the code. Corollary 3.2 gives the probability (on the choice of α) that the distance to the code reduces by folding.

During the query phase, the bad thing that can happen is that the prover cheated on only some values of a folding and that the verifier doesn't check these values. Proposition 3.3 gives the probability for the verifier not to detect an error during the query phase.

3.1.1 Commit phase soundness

With $f : \mathcal{D}_i \to \mathbb{F}$, $V_i = \mathsf{RS}[\mathcal{D}_i, \rho]$ and $V_{i+1} = \mathsf{RS}[\mathcal{D}_{i+1}, \rho]$, we want to bound

$$\mathop{\mathbb{P}}_{\alpha\in\mathbb{F}}\big(\Delta(\mathbf{Fold}[f,\alpha], V_{i+1}) \leqslant \Delta(f, V_i) - \varepsilon\big).$$

Theorem 3.1 will be used by contrapositive to show that if two words don't coincide with codewords on a same big enough support, then there are not a lot of α's that make their combination get closer to the code than what they were both.

It is in Theorem 3.1 that the constraint $\delta \leqslant \delta_{\max}$ appears. The next results will inherit this hypothesis. For this section we will set $\delta_{\max} := J_\varepsilon(J_\varepsilon(\Delta(V_0)))$, with ε a parameter we can choose. The Johnson bound is used twice in the proof of Theorem 3.1, so $\delta_{\max}$ is a composition of twice the Johnson function.

Theorem 3.1 [BKS18]

Let $V \subseteq \mathbb{F}^n$ be a linear code. Let $u^*, u \in \mathbb{F}^n$ and $\varepsilon, \delta > 0$ such that $\delta \leqslant \delta_{\max}$.
Let $A = \{\alpha \in \mathbb{F} \mid \Delta(u^* + \alpha u, V) < \delta - \varepsilon\}$.
If $|A| > \frac{2}{\varepsilon^3}$ then there exists $T \subseteq [\![1, n]\!]$ such that $u_{|T} \in V_{|T}$, $u^*_{|T} \in V^*_{|T}$ and $|T| > (1-\delta)n$.

Proof.
For $\alpha \in A$, denote by v^α an element of V such that $\Delta(u^* + \alpha u, v^\alpha) < \delta - \varepsilon$, i.e. $\Delta(u^*, v^\alpha - \alpha u) < \delta - \varepsilon$.
Consider the graph $G = (A, E)$ where

$$E = \{(\alpha, \alpha') \mid \Delta(v^\alpha - \alpha u, v^{\alpha'} - \alpha' u) < J_\varepsilon^{-1}(\delta)\},$$

and $\mathrm{co}G = (A, A^2 \setminus E)$ its complement graph.
Let $S_A \subseteq A$ be a clique of $\mathrm{co}G$ and $S = \{v^\alpha - \alpha u \mid \alpha \in S_A\}$. Since S_A is a clique of $\mathrm{co}G$, we have $\Delta(S) \geqslant J_\varepsilon^{-1}(\delta)$, so by applying the Johnson bound we get that S is $(\delta, \frac{1}{\varepsilon})$-list decodable, so

$$|B(u^*, \delta) \cap S| \leqslant \frac{1}{\varepsilon}. \tag{2}$$

But since $\Delta(u^*, v^\alpha - \alpha u) < \delta - \varepsilon$ by definition of v^α, we get that $B(u^*, \delta) \cap S = S$. So $|S_A| \leqslant \frac{1}{\varepsilon}$. Therefore $\mathrm{co}G$ doesn't have any clique of size $> \frac{1}{\varepsilon}$. We are now going to use Turán's theorem.

Theorem Turán's theorem

Every graph with n vertices that does not contain a clique of size $r+1$ has at most $(1 - \frac{1}{r})\frac{n^2}{2}$ edges.

By applying Turán's theorem on $\mathrm{co}G$, we get that G has at least $\varepsilon\frac{|A|(|A|-1)}{2}$ edges. Therefore $\sum_{\alpha \in A} \deg_G(\alpha) = \varepsilon|A|(|A|-1)$ so there exists $\alpha_0 \in A$ such that $\deg_G(\alpha_0) \geqslant \varepsilon|A| - 1$. Let

$$B = \{\alpha \in A \mid (\alpha_0, \alpha) \in E\}.$$

Then $|B| \geqslant \varepsilon|A| - 1$ and for all $\alpha \in B$,

$$\Delta(v^{\alpha_0} - \alpha_0 u, v^\alpha - \alpha u) = \Delta\left(u, \frac{1}{\alpha - \alpha_0}(v^\alpha - v^{\alpha_0})\right) < J_\varepsilon^{-1}(\delta).$$

Let $C_v = \left\{\alpha \in B \mid v = \frac{1}{\alpha_0 - \alpha}(v^\alpha - v^{\alpha_0})\right\}$ and $V_u = V \cap B(u, J_\varepsilon(\lambda))$. Then B is the disjoint union $\bigsqcup_{v \in V_u} C_v$, so $|B| = \sum_{v \in V_u} |C_v|$. Therefore there exists $v \in V_u$ such that $|C_v| \geqslant \frac{1}{|V_u|}|B|$. By Johnson's bound, we have that

$$|B(u, J_\varepsilon^{-1}(\delta)) \cap V| \leqslant \frac{1}{\varepsilon}, \tag{3}$$

so $|C_v| \geqslant \varepsilon|B|$. For all $\alpha \in C_v$, $v = \frac{1}{\alpha - \alpha_0}(v^\alpha - v^{\alpha_0})$, so with $v^* = v^{\alpha_0} - \alpha_0 v$, we have that for all $\alpha \in C_v$, $v^\alpha = v^* + \alpha v$. So by definition of v^α, for all $\alpha \in C_v$, $\Delta(u^* - v^*, \alpha(v - u)) < \delta - \varepsilon$.

Let $\overline{T} = \{i \in [\![1, n]\!] \mid (u_i^*, u_i) \neq (v_i^*, v_i)\}$. Let $D_\alpha = \{i \in \overline{T} \mid u_i^* - v_i^* = \alpha(v_i - u_i)\}$. For $i \in \overline{T}$ there can be at most one $\alpha \in C_v$ such that $u_i^* - v_i^* = \alpha(v_i - u_i)$ ($\frac{u_i^* - v_i^*}{v_i - u_i}$ if $v_i \neq u_i$, and none if $v_i = u_i$ and $v_i^* \neq u_i^*$). I.e. for $i \in \overline{T}$ there is at most one $\alpha \in C_v$ such that $i \in D_\alpha$.

So we have $\sum_{\alpha \in C_v} |D_\alpha| \leqslant |\overline{T}|$ and there exists α_1 such that $|D_{\alpha_1}| \leqslant \frac{|\overline{T}|}{|C_v|} \leqslant \frac{n}{|C_v|}$. If $i \in \overline{T}$, and if $i \notin D_{\alpha_1}$, we have that $u_i^* - v_i^* \neq \alpha_1(v_i - u_i)$. Thus with $T = [\![1, n]\!] \setminus \overline{T}$,

$$\delta - \varepsilon > \Delta(u^* - v^*, \alpha_1(v - u)) \geqslant \underbrace{\underbrace{1 - \frac{|T|}{n}}_{i \notin T} - \frac{|D_{\alpha_1}|}{n}}_{i \notin D_{\alpha_1}} \geqslant 1 - \frac{|T|}{n} - \frac{1}{|C_v|}.$$

Since $|C_v| \geqslant \varepsilon|B| \geqslant \varepsilon(\varepsilon|A| - 1) \geqslant \frac{1}{\varepsilon}$, $|T| > n\left(1 - \delta + \varepsilon - \frac{1}{|C_v|}\right) \geqslant (1-\delta)n$.

This is the proof from [BKS18] on which I added necessary details. Note that the Johnson bound is used only to get Equations (2) and (3). In section 3.2 we will take as hypothesis Equations (2) and (3) with more general parameters than $1/\varepsilon$ and δ but still centered in u^* and u, hoping that we will be able to prove that hypothesis for u^* and u.

This important result can now be applied in the context of the FRI protocol to get the "commit soundness".

Corollary 3.2 Commit soundness [BKS18]

Let $f : \mathcal{D}_i \to \mathbb{F}$ be an arbitrary function. Let $\varepsilon > 0$ and $\delta = \min\left(\Delta\left(f, V_i\right), \delta_{\max}\right)$. Then

$$\mathbb{P}_{\alpha \in \mathbb{F}}\left(\Delta\left(\mathbf{Fold}[f,\alpha], V_{i+1}\right) \leqslant \delta - \varepsilon\right) \leqslant \frac{2}{\varepsilon^3|\mathbb{F}|}.$$

Proof.
We use Theorem 3.1 by contrapositive with $A = \{\alpha \in \mathbb{F} \mid \Delta(\mathbf{Fold}[f,\alpha], V_{i+1}) \leqslant \delta - \varepsilon)\}$.
If by contradiction $|A| > \frac{2}{\varepsilon^3}$, then there exists $Q_1, Q_2 \in V_{i+1}$ and a subset $T \subseteq \mathcal{D}_{i+1}$ as in Theorem 3.1.
Then with $R(X) = Q_1(X^2) + XQ_2(X^2)$ we have

$$\deg(R) \leqslant 2\max(\deg(Q_1), \deg(Q_2)) + 1 \leqslant 2^{k-i} - 1 < 2^{k-i},$$

so $R \in V_i$, and thus $\Delta(f, V_i) \leqslant 1 - \frac{|T|}{|\mathcal{D}_{i+1}|} < \delta$ contradicts our assumption.

3.1.2 Query phase soundness

Now that we have the soundness for the commit phase, we can compute the soundness for the query phase assuming that the functions committed don't get closer to the code, i.e. that there is no commit error. This is thanks to the formula of total probability:

$$\begin{aligned}\mathbb{P}(V \text{ accepts}) &= \mathbb{P}(V \text{ accepts} \mid \text{commit error})\mathbb{P}(\text{commit error}) \\ &\quad + \mathbb{P}(V \text{ accepts} \mid \overline{\text{commit error}})\mathbb{P}(\overline{\text{commit error}}) \\ &\leqslant \mathbb{P}\left(\text{commit error}\right) + \mathbb{P}\left(V \text{ accepts} \;\middle|\; \overline{\text{commit error}}\right).\end{aligned}$$

Proposition 3.3 Query soundness [BKS18]

Let $\varepsilon > 0$, $m \in \mathbb{N}^*$, $\delta := \Delta(f_0, V_0)$ and r such that $2^r = \rho|\mathcal{D}_0|$.
Assuming that for all i, $\Delta(\mathbf{Fold}[f_i, x_i], V_{i+1}) > \delta_i - \varepsilon$, the probability not to detect an error by doing m iterations of the query phase is at most $(1 - \min(\delta, \delta_{\max}) + r\varepsilon)^m$.

Now, combining Corollary 3.2 and Proposition 3.3 we get the soundness of the whole protocol.

Theorem 3.4 FRI protocol soundness [BKS18]

Let $\varepsilon > 0$, $l \in \mathbb{N}^*$, $\delta := \Delta(f_0, V_0)$ and $r \in \mathbb{N}$ such that $2^r = \rho|\mathcal{D}_0|$.
When the FRI protocol is invoked on $f_0 : \mathcal{D}_0 \to \mathbb{F}$ with m iterations of the query phase, if $\delta := \Delta(f_0, V_0) > 0$ then the verifier accepts with probability at most

$$\frac{2r}{\varepsilon^3|\mathbb{F}|} + \left(1 - \min(\delta, \delta_{\max}) + r\varepsilon\right)^m,$$

where the probability is taken on the α_i's and the s_0's.

The reworked proof of both Corollary 3.2 and Theorem 3.4 is detailled in Appendix A.1.1.

The parameters we can adjust to get the required soundness are ε, $|\mathbb{F}|$ and m. m adds interaction, which makes the proof longer, ε must be big in the commit soundness and must be small in the query soundness, and $\mathbb{F}$ must be very big to counter ε^3 but it makes all the computations more expensive. Having a larger $\delta_{\max}$ allows to reduce the query soundness and therefore to reduce the size of the proof.

3.2 Generalization

The idea is to revisit the proofs of Theorem 3.1 and Corollary 3.2 but to remove the use of the Johnson bound, which is only used on the specific words u^* and u, by a "local list-decodability".

First, here is the most general generalization of Theorem 3.1. But the hypothesis Equation (4) is too unconvenient because it is not applied to the code but to a set S that is not totally arbitrary. Corollary 3.7 below is a simpler version.

Theorem 3.5 Generalization of Theorem 3.1 (inspired from [BKS18])

Let $V \subseteq \mathbb{F}^n$ be a linear code. Let $\varepsilon, \delta, \gamma^*, \gamma > 0$ such that $\delta \leqslant \gamma^* + \varepsilon$. Let $\mu, \mu^* \in \mathbb{N}^*$. Let $u^*, u \in \mathbb{F}^n$.
Let $A = \{\alpha \in \mathbb{F} \mid \Delta(u^* + \alpha u, V) < \delta - \varepsilon\}$ and let $v^\alpha \in V$ be the closest element to $u^* + \alpha u$.
Suppose that

$$\forall S \subseteq \{v^\alpha - \alpha u \,|\, \alpha \in A\}, \text{ if } \Delta(S) \geqslant \gamma \text{ then } |B(u^*, \gamma^*) \cap S| \leqslant \mu^* \tag{4}$$

$$|B(u, \gamma) \cap V| \leqslant \mu. \tag{5}$$

If $|A| > \frac{2\mu\mu^*}{\varepsilon}$ then there exists $T \subseteq [\![1, n]\!]$ such that $u_{|T} \in V_{|T}$, $u^*_{|T} \in V^*_{|T}$ and $|T| > (1 - \delta)n$.

The proof is exactly the same as for Theorem 3.1 and a detailled version is available in Appendix A.1.2. To see that this result is indeed a generalization we can prove Theorem 3.1 with it.

Example 3.6 Theorem 3.1 by application of Theorem 3.5

Let $\delta, \varepsilon > 0$ such that $\delta \leqslant J_\varepsilon(J_\varepsilon(\lambda))$. Let $\gamma = J_\varepsilon^{-1}(\delta)$, $\gamma^* = \delta$, $\mu = \mu^* = \frac{1}{\varepsilon}$. We have that $\delta \leqslant \gamma^*$. For all $S \subseteq \mathbb{F}^n$ such that $\Delta(S) \geqslant J_\varepsilon^{-1}(\delta)$, we have $|B(u^*, \delta) \cap S| \leqslant \frac{1}{\varepsilon}$ by Johnson bound. Also by Johnson bound, $|B(u, J_\varepsilon(\lambda) \cap V| \leqslant \frac{1}{\varepsilon}$. Since $J_\varepsilon^{-1}(\delta) \leqslant J_\varepsilon(\lambda)$, $|B(u, J_\varepsilon^{-1}(\delta)) \cap V| \leqslant |B(u, J_\varepsilon(\lambda)) \cap V| \leqslant \frac{1}{\varepsilon}$.

Equation (4) replaces the use of the Johnson bound to get Equation (2) in Theorem 3.1 and Equation (5) replaces the Johnson bound to get Equation (3). However Equation (4) is not applied on the code but to a set S that won't have the required properties. For this one it seems that the Jonhson bound will be the best we will achieve.

Corollary 3.7 Theorem 3.5 with only one Johnson

Let $V \subseteq \mathbb{F}^n$ be a linear code. Let $\varepsilon, \delta, \gamma > 0$ such that $\delta \leqslant \delta_{\max} := J_\varepsilon(\gamma) + \varepsilon$. Let $\mu \in \mathbb{N}^*$. Let $u^*, u \in \mathbb{F}^n$.
Let $A = \{\alpha \in \mathbb{F} \mid \Delta(u^* + \alpha u, V) < \delta - \varepsilon\}$. Suppose that

$$|B(u, \gamma) \cap V| \leqslant \mu. \tag{6}$$

If $|A| > \frac{2\mu}{\varepsilon^2}$ then there exists $T \subseteq [\![1, n]\!]$ such that $u_{|T} \in V_{|T}$, $u^*_{|T} \in V^*_{|T}$ and $|T| > (1 - \delta)n$.

In Theorem 3.5 we see that we have $\delta_{\max} = \gamma^* + \varepsilon$, and in Corollary 3.7 we have $\delta_{\max} = J_\varepsilon(\gamma) + \varepsilon$, which won't be better than the state of the art [BCI+20].

Lemma 3.8 Commit soundness, generalization (inspired from [BKS18])

Let $f : \mathcal{D}_i \to \mathbb{F}$, μ and $\delta, \varepsilon, \gamma > 0$ such that $\delta = \Delta(f, V_i)$. Let f_0 and f_1 be the even and odd parts of f. Let $A = \{\alpha \in \mathbb{F} \mid \Delta(\textbf{Fold}[f, \alpha], V_{i+1}) < \delta - \varepsilon\}$ and suppose that Equation (6) of Corollary 3.7 is satisfied with $u := f_1$.
Then

$$\underset{x \in \mathbb{F}}{\mathbb{P}}(\Delta(\textbf{Fold}(f, x), V_{i+1}) < \delta - \varepsilon) \leqslant \frac{\mu}{\mathbb{F}}.$$

This result inherits the $\delta_{\max}$ of Corollary 3.7. The proof is exactly the same as for Corollary 3.2, replacing ε by $\frac{1}{\mu}$, and is also detailled in Appendix A.1.2.

Lemma 3.8 is compatible with Proposition 3.3, which gives the soundness of the whole protocol:

$$\frac{2\mu r}{\varepsilon^2|\mathbb{F}|} + (1 - \min(\delta, \delta_{\max}) + r\varepsilon)^m .$$

Since the application to IRS would not have improved $\delta_{\max}$ further than [BCI+20] I did not push further this direction. Moreover it appeared at this point that it would not have been possible to use IRS because there is no randomness on word for which we want Equation (6). Therefore I worked on another approach of the FRI protocol, which is only a bit more complex, and in which it is possible to completely replace the use of any Johnson bound.

4 DEEP method

The DEEP-FRI protocol was introduced in [BGKS20]. DEEP is an acronym for Domain Extending for Eliminating Pretenders. The DEEP-FRI protocol is an IOPP protocol for Reed-Solomon codes that aims to improve the soundness of the FRI protocol by evaluating the functions outside of the evaluation set.

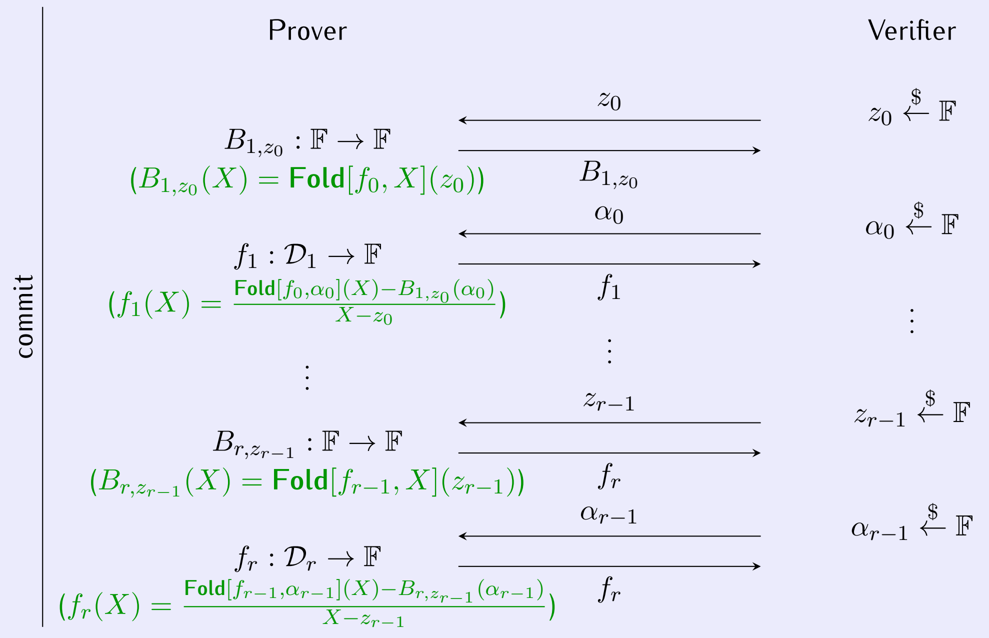
The advantage of this protocol over the FRI is that the word on which the Johnson bound is applied depends on some randomness of the verifier. In the previous part, the Johnson bound was applied on words u^* and u chosen by the prover, but here it will be applied on $u^* + \alpha u$ with α chosen randomly by the verifier. It gives two advantages: the first is that the Johnson bound is used only once, and the second is that having randomness of the verifier makes the hypothesis we are doing on $u^* + \alpha u$ (Hypothesis 4.5) is a kind of probabilistic list-decoding, which is what we are expecting from interleaved codes (see Section 4.3). Unfortunately we will see that I haven't been able to find interesting results using the properties of IRS by using so few randomness.

4.1 The DEEP-FRI protocol [BGKS20]

Only the commit phase changes between the FRI and DEEP-FRI protocol. The idea is to be able to evaluate the functions out of their domains $\mathcal{D}_i$.

Definition 4.1 **Commit phase of the DEEP-FRI protocol [BGKS20]**

The verifier sends a random element z of the field so that the prover commits a folding $B_z(X) :=$ **Fold**$[f, X](z)$, then the verifier sends another random element α to fold around, and the prover commits the new function derived from the folding, $f(X) := \frac{\textbf{Fold}[f,\alpha](X) - B_z(\alpha)}{X - z}$.


commit
Prover
Verifier
z_0
$z_0 \overset{\$}{\leftarrow} \mathbb{F}$
$B_{1,z_0} : \mathbb{F} \to \mathbb{F}$
$(B_{1,z_0}(X) = \textbf{Fold}[f_0, X](z_0))$
B_{1,z_0}
α_0
$\alpha_0 \overset{\$}{\leftarrow} \mathbb{F}$
$f_1 : \mathcal{D}_1 \to \mathbb{F}$
$(f_1(X) = \frac{\textbf{Fold}[f_0,\alpha_0](X) - B_{1,z_0}(\alpha_0)}{X - z_0})$
f_1
$\vdots$
z_{r-1}
$z_{r-1} \overset{\$}{\leftarrow} \mathbb{F}$
$B_{r,z_{r-1}} : \mathbb{F} \to \mathbb{F}$
$(B_{r,z_{r-1}}(X) = \textbf{Fold}[f_{r-1}, X](z_{r-1}))$
f_r
α_{r-1}
$\alpha_{r-1} \overset{\$}{\leftarrow} \mathbb{F}$
$f_r : \mathcal{D}_r \to \mathbb{F}$
$(f_r(X) = \frac{\textbf{Fold}[f_{r-1},\alpha_{r-1}](X) - B_{r,z_{r-1}}(\alpha_{r-1})}{X - z_{r-1}})$
f_r

The query phase works the same way but the verification of the functions f_i's doesn't have the same expression.

query

$$\left.\begin{array}{c}\text{Verifier chooses } s_0 \overset{\$}{\leftarrow} \mathcal{D}_0\text{, defines } s_{i+1} := s_i^2 \text{ and accepts if}\\ \mathsf{Fold}[f_0,\alpha_0](s_1) = (s_{i+1}-z_i)f_{i+1}(s_{i+1}) + B_{i+1,z_i}(\alpha_i)\\ \vdots \\ \mathsf{Fold}[f_{r-1},\alpha_{r-1}](s_r) = (s_r - z_{r-1})f_r(s_r) + B_{r,z_{r-1}}(\alpha_{r-1}).\end{array}\right\}\text{repeat}$$

The equivalent of Theorem 3.1 and Corollary 3.2 is the following theorem.

Notation 4.2

For $z, b \in \mathbb{F}$, let $V^{z,b} := \{f \in V \mid f(z) = b\} \subseteq V$ be the subcode where codewords evaluate to b on z.

Theorem 4.3 DEEP method for RS codes [BGKS20]

Let $V = \mathsf{RS}[\mathcal{D},\rho]$. For $\delta > 0$, let $\mu_\delta \in \mathbb{N}$ such that V is (δ,μ_δ)-liste-decodable. Let $u^*, u \in \mathbb{F}^{\mathcal{D}}$. Let $\delta > 0$ and $0 < \varepsilon < \frac{1}{3}$. Suppose that

$$\underset{x,z\in\mathbb{F}}{\mathbb{P}}\left(\Delta(u^* + xu, V^{z,B_z(x)}) < \delta\right) \geqslant 2\mu_\delta \cdot \left(\frac{\rho|\mathcal{D}|}{|\mathbb{F}|} + \varepsilon\right)^{1/3} + \frac{4}{\varepsilon^2|\mathbb{F}|}. \tag{7}$$

Then there exists $T \subseteq \mathcal{D}$ such that $u^*_{|T} \in V_{|T}$, $u_{|T} \in V_{|T}$ and $|T| > (1-\delta-\varepsilon)|\mathcal{D}|$.

The proof of a more general result (Proposition A.4) is detailled in Appendix A.2. Equation (7) gives the commit phase soundness for the DEEP-FRI protocol. The right term looks like the soundness of Corollary 3.2. Notice that there is only one Johnson bound used here, so there is an ε^2 at the denominator whereas there is an ε^3 for Corollary 3.2. The first term of Equation (7) is used to create a line in some space to extend the evaluation space of the words. It requires taking 3 random points, so the probability will be cubed and there is the cube root here.

4.2 Application to IRS

The DEEP-FII protocol (DEEP-Fast IRS IOPP) will be exactly the DEEP-FRI protocol, except that the z_i's and α_i's are taken in $\mathbb{F}^\ell$, the functions $B_{i,z}$ are $\mathbb{F}^\ell \to \mathbb{F}^\ell$ and the f_i are $\mathcal{D}_i \to \mathbb{F}^\ell$ with $\mathcal{D} \subseteq \mathbb{F}$ (see Definition B.1). I worked with Sarah Bordage to make sure that the protocol is correct.

Definition 4.4 Interleaved folding

Let $f : \mathbb{F}^* \to \mathbb{F}^\ell$ and $\alpha = \begin{pmatrix}\alpha_1\\ \vdots \\ \alpha_\ell\end{pmatrix} \in \mathbb{F}^\ell$. Let $f_1, \ldots, f_\ell : \mathbb{F}^* \to \mathbb{F}$ such that $f = \begin{pmatrix} f_1 \\ \vdots \\ f_\ell\end{pmatrix}$. Then

$$\mathsf{Fold}[f,\alpha] = \begin{pmatrix}\mathsf{Fold}[f_1,\alpha_1]\\ \vdots \\ \mathsf{Fold}[f_\ell,\alpha_\ell]\end{pmatrix}.$$

The result we will need to replace the Johnson bound is something like the following. We want to have a value p and $\delta_{\max}$ as high as possible, and $\mu \in \mathbb{N}$ as low as possible, maybe $\mu = 1$, such that the following holds.

Hypothesis 4.5 $(\mu, p, \delta_{\max})$-hypothesis for probabilistic list-decoding on an arbitrary line of an IRS

Let $\mu \in \mathbb{N}$ and $p, \delta_{\max} \in [0,1]$. Let $V = \mathsf{IRS}[\mathcal{D},\rho,\ell] \subseteq \mathbb{F}^{n\times\ell}$. Let $u^*, u \in \mathbb{F}^{n\times\ell}$ be arbitrary points of the ambiant space. Then

$$\mathbb{P}_{\alpha\in\mathbb{F}}\big(|B(u^*+\alpha u,\delta_{\max})\cap V|\leqslant\mu\big)\geqslant p. \tag{8}$$

Note that we can easily have $p=1$ with $\mu:=\frac{1}{\varepsilon}$ and $\delta_{\max}:=J_\varepsilon(1-\rho)\approx 1-\sqrt{\rho}$ the Johnson bound, or even $p=1$ and $\mu=1$, with $\delta_{\max}:=\frac{1}{2}(1-\rho)$ the deterministic unique decoding radius. But we want to have δ as close to $1-\rho$ as possible. We can also have $\delta_{\max}=1-\rho$ but then μ becomes exponential in the size of $\mathcal{D}$, which makes the soundness explode. Here we are targetting $\delta_{\max}:=\frac{\ell}{\ell+1}(1-\rho)$, because results like Proposition 2.4 suggests that it should provide satisfying results.

The application of Theorem 4.3 to IRS will be the following.

Proposition 4.6 **DEEP method for Interleaved Reed-Solomon codes with probability on the list bound**

Let $\mu\in\mathbb{N}$, $p,\delta_{\max}\in[0,1]$, and suppose that Hypothesis 4.5 is valid for $(\mu,p,\delta_{\max})$. Let $V=\mathrm{IRS}[\mathcal{D},\rho,\ell]$. Let $u^*,u\in\mathbb{F}^{\mathcal{D}\times\ell}$, $0<\delta\leqslant\delta_{\max}$ and $0<\varepsilon\leqslant\frac{1}{3}$. Denote $\eta=2\mu\cdot\left(\frac{\rho|\mathcal{D}|}{|\mathbb{F}|}+\varepsilon\right)^{1/3}+\frac{4}{\varepsilon^2|\mathbb{F}|}$ the same probability as in Equation (7), with μ generalizing μ_δ. Suppose that

$$\mathbb{P}_{\alpha\in\mathbb{F},z\in\mathbb{F}}\big(\Delta(u^*+\alpha u,V^{z,B_z(\alpha)})<\delta\big)\geqslant p\eta+(1-p). \tag{9}$$

Then there exists $T\subseteq\mathcal{D}$ such that $u^*_{|T}\in V_{|T}$, $u_{|T}\in V_{|T}$ and $|T|>(1-\delta-\varepsilon)|\mathcal{D}|$.

As for Theorem 4.3, the proof of Proposition A.4, a more general result, is detailled in Appendix A.2. Equation (9) justifies that we want to have p as close to 1 as possible. If $p=1$ then we get η as in Theorem 4.3, and if $p=0$ then we get 1 which is useless. The soundness theorem associated is also almost the same as in [BGKS20], so I have not worked on the proof.

Theorem 4.7 **Soundness theorem for the DEEP-FII protocol (almost [BGKS20])**

Let $\mu\in\mathbb{N}$, $p,\delta_{\max}\in[0,1]$, and suppose that Hypothesis 4.5 is valid for $(\mu,p,\delta_{\max})$. Then, when the DEEP-FII protocol is invoked on oracle $f_0:\mathcal{D}_0\to\mathbb{F}^\ell$, with m iterations of the query phase, if $\delta=\Delta(f_0,V_0)>0$ then the verifier accepts with probability at most

$$\underbrace{(p\eta+1-p)\log|\mathcal{D}_0|}_{\text{commit soundness}}+\underbrace{(1-\min(\delta,\delta_{\max})+(\log|\mathcal{D}_0|-2r)\varepsilon)^m}_{\text{query soundness}},$$

with $\eta=2\mu\cdot\left(\frac{\rho|\mathcal{D}_0|}{|\mathbb{F}|}+\varepsilon\right)^{1/3}+\frac{4}{\varepsilon^2|\mathbb{F}|}$ and where the probability is taken on the z_i's, α_i's and s_0's.

4.3 Hypothesis 4.5

I have not stated Hypothesis 4.5 as a result I have not been able to prove it for interesting values of $\mu,p,\delta_{\max}$. I worked with Daniel Augot, Ilaria Zappatore and Sarah Bordage to try to get such a result.

The first approach was to start from Proposition 2.4 adapt the proof to our case, but it strongly uses the fact that the error columns must be uniformly random. There is not enough randomness in the protocol. There are only ℓ random bits at each step, which doesn't seem te be enough. Having $|\mathcal{D}|$ random bits would work but $\mathcal{D}$ is really huge in practice, making such a protocol completely useless because of the communication complexity.

Another approach was to say that the columns of the matrix $u^*+\alpha u$ will behave almost randomly, and therefore the non null columns of the difference between $u^*+\alpha u$ and the closest codeword would be linearly independent with a good probability, in order to apply the two following results.

Theorem 4.8 **Correction of interleaved codes with linearly independent error columns [MK90]**

Let $V \subseteq \mathbb{F}^{n\times\ell}$ be a linear code of minimal distance d. If the code is seen as a code over $\mathbb{F}^\ell$, then any pattern of $d-2$ or less errors can be corrected if the error columns are linearly independent.

But unfortunalety this cannot work because the prover can commit two words that will never form a line of words with linearly independent error columns, as shown in the following counterexample.

Example 4.9 **Counterexample**

Suppose that the prover commits words $u = v + e$ and $u^* = v^* + e$, with $v, v^* \in V$ and $e \in \mathbb{F}^{\ell\times n}$, where the non-null columns of e are linearly dependent. Then $\forall \alpha \in \mathbb{F}, u_x = v^* + \alpha v + (1+\alpha)e = v_x + (1+\alpha)e$. So the "error" of $u^* + \alpha u$ has linearly dependent non-null columns with probability 1 over the choice of α.

A last idea, but that I have not explored, is to only consider the functions that the prover can commit, i.e. the functions that can be obained by arithmetization. We already know that these functions have a gap promise property (Lemma 1.11), so maybe there are other interesting properties, and restricting Hypothesis 4.5 to u^* and u obtained by arithmetization allows to get more interesting results.

Part III

De Bruijn graphs for arithmetization

5 Circuits to De Bruijn graphs [Spi95]

5.1 Colored wrapped de Bruijn graphs to represent circuits

The use of De Bruijn graphs to represent circuits was introduced in [Spi95]. I first thought that their use was similar in [BBHR18b], so I studied this, but in fact it is completely different. Nevertheless, it makes a good introduction to De Bruijn graphs and the arithmetization is interesting.

Definition 5.1 **Wrapped De Bruijn graph**

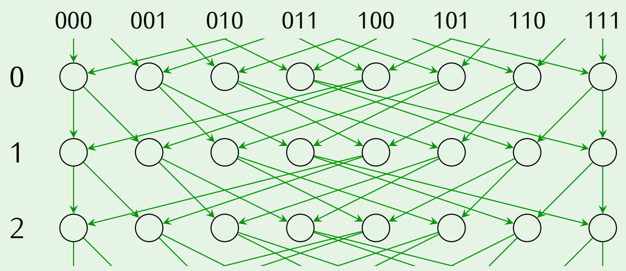
The wrapped De Bruijn graph (or simply De Bruijn graph) of parameter n is the directed graph $B_n = (V_n, E_n)$ where

$$V_n = \{(i, (x_1, ..., x_n)) \mid i \in \mathbb{Z}/n\mathbb{Z}, x_i \in \{0,1\}\},$$

and the edges are $(i, (x_1, ..., x_n)) \to (i+1, (x_2, ..., x_n, b))$ with $b \in \{0,1\}$.

Example 5.2 **De Bruijn graph**

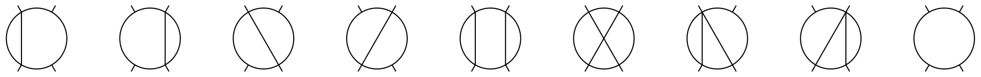
Here is the wrapped De Bruijn graph of size $n = 3$.



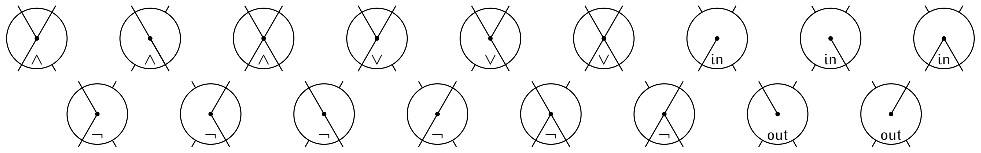
An interesting property is that there is exactly one path of length n from a node $(0, (x_1, ..., x_n))$ to any node $(0, (y_1, ..., y_n))$. There are two edges going out from a node, so a bitstring can denote a path in the graph. For $b \in \{0, 1\}$, by denoting b the edge $(i, (x_1, ..., x_n)) \to (i + 1, (x_2, ..., x_n, x_1 \oplus b)$, the path from $(x_1, ..., x_n)$ to $(y_1, ..., y_n)$ is $(x_1 \oplus y_1, ..., x_n \oplus y_n)$.

A wrapped De Bruijn graph of size n is able to represent a circuit of $\log_2 n$ gates, including additional gates for the inputs and the output. But we have to specify a routing that represents the given circuit. To make an analogy, a circuit can be easily embedded in a complete binary tree with some leaves representing the inputs, some nodes representing gates and the root representing the output. De Bruijn graphs are like complete binary trees and we have to specify which nodes correspond to which gate, and what is the effective connection between them.

We want to represent the circuit as a coloring of the graph. So the gates and routing will be represented as colors. So it is not the edges that will be marked, but the nodes. There is one line, the line $i = 0$, that will be used to represent the gates. The other nodes are only used for routing. Every node has two inputs and two outputs, so the possible connections for the routing nodes are the 9 following.



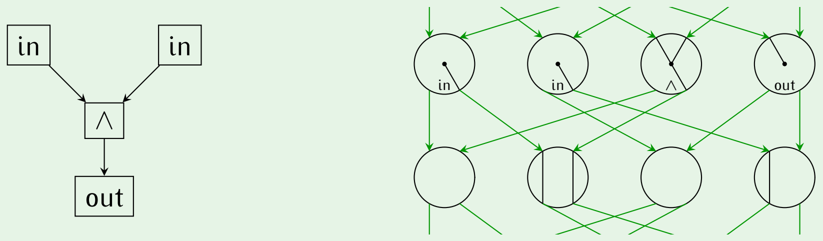
The nodes representing a gate also have to represent their routing. All routings are not possible for each gate (the gates have 0, 1 or 2 inputs for example). The possible coloring are 15 the following.



Thus an instance of circuit can be represented as a 24 colors coloring of a wrapped De Bruijn graph.

Example 5.3 **Instance of circuit as De Bruijn graph**

Here are a circuit and the associated colored De Bruijn graph.



Now we also want to represent the witness. The witness of satisfiyability could in theory be the values of the inputs, but checking that the witness is valid would require to go through all the circuit, which we want to avoid. Instead we want the witness to be a proof, so we will also represent the witness as a coloring telling for each node which entries or outputs are mapped to `true` and which entries or outputs are mapped to `false`. So a witness of satisfyability of a circuit can be represented as a 2^4 colors coloring of a De Bruijn graph. Given an instance and a witness we will only check local consistency on random nodes to be convinced of the validity of the witness.

5.2 De Bruijn graphs arithmetization

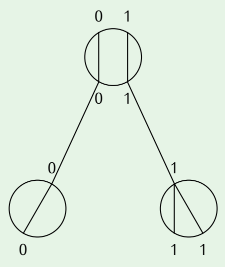
For arithmetization we see the graph B_n as a subset of $A \times \mathbb{F}_{2^{n/2}} \times \mathbb{F}_{2^{n/2}}$ where $A = \{\alpha^i \mid 0 \leqslant i \leqslant n-1\}$ and α is a generator of the multiplicative group of $\mathbb{F}_{2^{n/2}}$. The node (t, x, y) has edges to $(\alpha t, y, \alpha x)$ and $(\alpha t, y, \alpha x + 1)$. x and y are swapped but this graph is isomorphic to the wrapped De Bruijn graph defined earlier. An instance is a coloring, so we see it as a function $A \times \mathbb{F}_{n/2} \times \mathbb{F}_{n/2} \to C_1$ where $C_1 \subseteq \mathbb{F}_{n/2}$ is a first set of colors. A solution is another coloring, so we also see it as a function $A \times \mathbb{F}_{n/2} \times \mathbb{F}_{n/2} \to C_2$ where $C_2 \subseteq \mathbb{F}_{n/2}$ is a second set of colors.

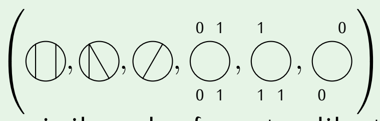
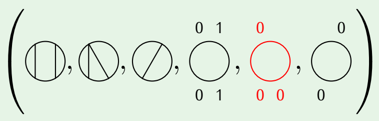
To check that a witness is valid, we want to check only locally. In order to check the validity of one node, we will check the consistency of the witness colors of the node with its two neighbors, with respect to their instance color. A tuple $(i_0, i_1, i_2, w_0, w_1, w_2) \in C_1^3 \times C_2^3$ is valid if a lot of rules are satisfies. A couple (i_j, w_j) respects validity if the inputs and outputs of the node (in w_j) respects the wiring. For a routing node it means that inputs are linked to outputs of same value (`true` or `false`), and for gate nodes it means that the inputs are linked to outputs with a value respecting the gate ($\wedge$, $\vee$ and $\neg$ are applied). A tuple (i_0, i_j, w_0, w_j) respects validity if, in the case where there is a wire between these nodes, if the output value of the first node is the same as the input value of the second node. If there is no wire between these nodes then it always respects validity.

With all these rules, it is possible to create a polynomial $\chi \in \mathbb{F}[X_0, X_1, X_2, Y_0, Y_1, Y_2]$ interpolating the valid values of colors (seen as elements of $\mathbb{F}$) to 0.

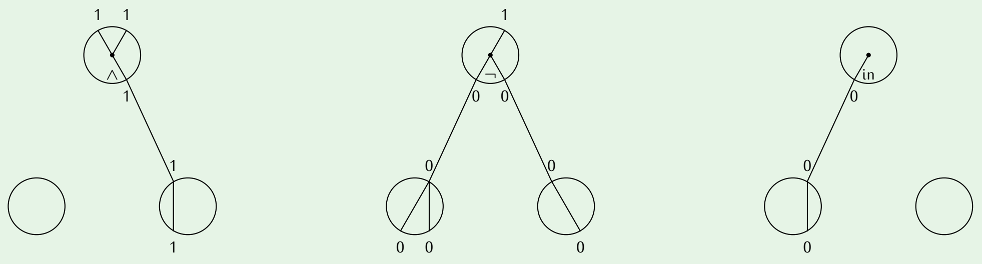
Example 5.4

These nodes are locally correct:



Therefore χ  $= 0$, but χ  $\neq 0$.

And there are similar rules for gates, like these ones:



6 De Bruijn graphs to check the memory consistency [BBHR18b]

With the arithmetization described in Section 1.2, the execution trace of a computation of T steps and using R registers has size RT. So if the computation has a high memory complexity ($R = \Omega(T)$) then the execution trace has size $O(T^2)$, which is too big. [BBHR18b] proposes another arithmetization such that the witness of the correct execution has always size $O(T \log T)$.

6.1 Checking the memory trace [Per17, BBHR18b]

The idea is to notice that at each step, only one register is accessed, so it is not necessary to consider all the registers at each step in the execution trace. Instead we consider a trace of the accesses to the memory. Like in Section 1.2 we consider a subgroup $H \subseteq \mathbb{F}^*$ of size $T+1$ and generator g.

Definition 6.1 **Memory trace**

A memory trace is represented by three functions:

- $l : H \to \{\text{read}, \text{write}, \text{ignore}\} \subseteq \mathbb{F}$ telling if the memory is read, written or ignored,
- $a : H \to \mathbb{F}$ giving the adress of the accessed cell, and
- $v : H \to \mathbb{F}$ giving the value that is read or written in the memory.

A memory trace is valid if, whenever a value is read in a cell, it corresponds to the value that was previously in the cell.

Definition 6.2 **Valid memory trace [Per17]**

A memory trace (l, a, v) is valid if for any $t \in [\![1, T]\!]$ such that $l(g^t) = \text{read}$, with $t_0 < t$ the maximum time such that $l(g^{t_0}) \neq \text{ignore}$ and $a(g^{t_0}) = a(g^t)$, we have $v(g^{t_0}) = v(g^t)$.

Checking the validity of a memory trace can done efficiently if the verifier knows of a permutation $\sigma : H \to H$ sorting the time steps by first the adress and second the time (i.e. $t \mapsto (a(\sigma(g^t)), \sigma(g^t))$ is an increasing function for the dictionnary order and any order on $\mathbb{F}$). With such a permutation we can construct a constraint polynomial $Q(X_1, X_2, X_3, Y_1, Y_2, Y_3)$ such that with $L(X), A(X), V(X)$ polynomials interpolating l, a, v on H and

$$P(X) := Q(L(\sigma(X)), A(\sigma(X)), V(\sigma(X)), L(g\sigma(X)), A(g\sigma(X)), V(g\sigma(X))), \tag{10}$$

we have that $\forall t \in [\![1, T]\!], P(g^t) = 0$ iff the memory trace is valid. For this we must have

$$Q(X_1, X_2, X_3, Y_1, Y_2, Y_3) = 0 \quad \text{iff} \quad (X_1 = \text{read} \wedge Y_1 \neq \text{ignore} \wedge X_2 = Y_2) \Rightarrow X_3 = Y_3.$$

The issue is that it is not possible to ensure that σ is a permutation and that the prover is not cheating with an arbitrary function $H \to H$. This is the subject of Section 6.2.

The memory trace, together with a sorting permutation, could be sufficient to check the validity of the memory, but it is impractical to check the validity of the execution. [Per17] gives the explicit construction of polynomial constraints for all the possible computation steps, and in particular the number of simultaneous register required, and [BBHR18b] claims that $R = O(\log T)$ registers suffices to have an execution trace that can be efficiently generated and verified. In this new arithmetization there are three types of constraints.

Definition 6.3 **Permuted execution trace constraints [BBHR18b]**

There are three types of constraints to check the validity of a computation:

- The local constraints.
- The execution constraints Q's, such that $\forall t, Q(\mathbf{r}(g^t), \mathbf{r}(g^{t+1})) = 0$.
- The memory constraints Q's, such that $\forall t, Q(\mathbf{r}(g^t), \mathbf{r}(\pi(g^t))) = 0$.

The memory constraints are written differently than in Equation (10), but with $\pi(x) = \sigma^{-1}(g\sigma(x))$ it is equivalent to test the consistency of all g^t with $\pi(g^t)$ and to check the consistency of all $\sigma(g^t)$ with $g\sigma(g^t)$.

6.2 De Bruijn graphs to algebraize permutations [BBHR18b]

Let $n \in \mathbb{N}$ such that $2^n = T + 1$. We saw in Section 5 that in a wrapped De Bruijn graph it is possible to go from a node $(0, (x_1, ..., x_n))$ to any node $(0, (y_1, ..., y_n))$ in n steps. In fact it is possible to use a routing coloring of a De Bruijn graph (without gate nodes) to represent a lot of functions $\{0, 1\}^n \to \{0, 1\}^n$, including the permutations. Furthermore, it is possible to encode the nodes of a De Bruijn graph in $\mathbb{F}$ in such a way that there are 8 affine functions such that the two neighbors of a node are its image by two of these functions.

Definition 6.4 **Algebraic De Bruijn graph [BBHR18b]**

Let $g_0 \in \mathbb{F}$ be a primitive element, $k := \lceil \log(n + 1) \rceil$, and ξ be a primitive binary polynomial of degree k. The node $(\overline{y_{k-1} ... y_0}^2, (x_0, ..., x_{n-1}))$ of the De Bruijn graph of size n is represented as

$$\sum_{j=0}^{n-1} x_j g_0^j + g_0^n \sum_{j=0}^{k-1} y_j g_0^j = x + g_0^n y.$$

Therefore the neighbors of the node $x + g_0^n y$ are the $v_b(x + g_0^n y)$ for $b \in \{0,1\}$, where

$$v_b(x + g^n y) = v_{x_{n-2}, y_{k-1}, b}(x + g_0^n y) \qquad v_{b_1,b_2,b_3}(z) := g_0 z + b_1(g_0^n + 1) + b_2(g_0^n \xi) + b_3.$$

Note that for all $b_1, b_2, b_3 \in \{0,1\}$, v_{b_1,b_2,b_3} is an affine function. Now we allow ourselves to use both writings undistinctly.

Theorem 6.5 gives a way to ensure that the function π given by the prover is a permutation.

Theorem 6.5 **Affine permutation [BBHR18b]**

Let (V_n, E_n) be the De Bruijn graph of size n. Let $c_0, c_1 : V_n \to C := \{g^t \mid 0 \leqslant t \leqslant T\}$ be colorings such that

1. $w \mapsto c_0(0, w)$ is injective,
2. $\forall w \in \{0,1\}^n, c_0(n-1, w) = c_1(n-1, w)$, and
3. for $b \in \{0,1\}$, $c_b(i, w)$ is equal to $c_b(v_0(i, w))$ or $c_b(v_1(i, w))$.

Then the function $\pi : C \to C$ induced by $\forall w, \pi(c_0(0, w)) = c_1(0, w)$ is a permutation.

Conversely, for any permutation $\pi : C \to C$, there exists colorings c_0 and c_1 satisfying the three properties above.

Let $c_0, c_1 : V_n \to C$ be the colorings induced by Theorem 6.5 for π. Let $C_0, C_1 \in \mathbb{F}[X]$ interpolating c_0, c_1 on V_n. By Theorem 6.5 we have that π is a permutation iff

1. $C_0(X) - C_*(X)$ cancels on $\{(i, w) \in V_n \mid i = 0\}$, with $C_*(X)$ interpolating a canonical bijection $c_* : \{(i, w) \in V_n \mid i = 0\} \to C$ which is known by the verifier,
2. $C_0(X) - C_1(X)$ cancels on $\{(i, w) \in V_n \mid i = n-1\}$, and
3. For $b \in \{0,1\}$, $\big(C_b(X) - C_b(v_0(X))\big)\big(C_b(X) - C_b(v_1(X))\big)$ cancels on V_n.

And Furthermore, we have

$$\forall t \in [\![1, T]\!], P(\mathbf{r}(g^t), \mathbf{r}(\pi(g^t))) = 0 \quad \text{iff} \quad \forall w \in \{0,1\}^n, P(\mathbf{r}(C_0(0, w)), \mathbf{r}(C_1(0, w))) = 0.$$

So we have constructed constraint polynomials to allow the verifier to efficiently check memory constraints.

For the verifier to be able to check the consistency of the proof polynomials efficiently, we required in Section 1.2 that the polynomial constraints cancel on $H = \{g^t \mid 0 \leqslant t \leqslant T\}$, so that the verifier can compute $\prod_{1 \leqslant t \leqslant T}(X - g^t) = X^{T+1} - 1$. Here with this construction the polynomials don't cancel on H, but the real arithmetization of [BBHR18b] makes the polynomials cancel on a convenient set.

References

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, page 46, 2018. `http://eprint.iacr.org/2018/046`.

[BCI$^+$20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed–Solomon Codes. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 900–909, 2020.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 31–60, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, 1991.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. 151:5:1–5:32, 2020. `https://eprint.iacr.org/2019/336`.

[BKS18] Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-Case to Average Case Reductions for the Distance to a Code. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference (CCC 2018)*, volume 102 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:23, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[BLNR22] Sarah Bordage, Mathieu Lhotel, Jade Nardi, and Hugues Randriam. Interactive Oracle Proofs of Proximity to Algebraic Geometry Codes. In Shachar Lovett, editor, *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:45, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with Polylog Query Complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

[MK90] J.J. Metzner and E.J. Kapturowski. A general decoding technique applicable to replicated file disagreement location and concatenated code decoding. *IEEE Transactions on Information Theory*, 36(4):911–917, 1990.

[Per17] Evgenya Pergament. Algebraic RAM. Master's thesis, Computer Science Department, Technion, 2017.

[Spi95] Daniel Alan Spielman. *Computationally efficient error-correcting codes and holographic proofs*. PhD thesis, Massachusetts Institute of Technology, 1995.

[Zap20] Ilaria Zappatore. *Primitivity of generalized translation based block ciphers*. PhD thesis, 2020.

Acknowledgements

I would like to deeply thank Daniel Augot to have proposed this really interesting subject and for having been able to bear my unclear explanations of what I understood of the papers I read.

I also would like to thank Ilaria and Sarah with whom I had the pleasure to work, and the whole GRACE team for their warm welcome and the general atmosphere that prevails all the time.

I hope to come back after my internship in Japan to do my PhD in the team, with Daniel Augot.

A Detailled proofs

A.1 FRI soundness proofs

A.1.1 Detailled proof of Theorem 3.4

Let $\delta_i = \min(J_\varepsilon(J_\varepsilon(\Delta(V_0))), \Delta(f_i, V_i))$. Let E_i be the bad event that $\Delta(\textbf{Fold}[f_i, \alpha_i], V_{i+1}) \leqslant \delta_i - \varepsilon$. Let accept be the event that the Verifier accepts.

By the formula of total probability,

$$\mathbb{P}(\text{accept}) = \underbrace{\mathbb{P}\left(\text{accept} \,\middle|\, \bigcup_{i=0}^{r-1} E_i\right)}_{\leqslant 1} \mathbb{P}\left(\bigcup_{i=0}^{r-1} E_i\right) + \mathbb{P}\left(\text{accept} \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right) \underbrace{\mathbb{P}\left(\bigcap_{i=0}^{r-1} \overline{E_i}\right)}_{\leqslant 1}$$

$$\leqslant \mathbb{P}\left(\bigcup_{i=0}^{r-1} E_i\right) + \mathbb{P}\left(\text{accept} \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right).$$

Corollary 3.2 implies that $\mathbb{P}(E_i) \leqslant \frac{2}{\varepsilon^3|\mathbb{F}|}$ so by the union bound we have that

$$\mathbb{P}\left(\bigcup_{i=0}^{r-1} E_i\right) \leqslant \frac{2r}{\varepsilon^3|\mathbb{F}|}.$$

With Q_j the event that the Verifier doesn't reject the j^{th} iteration of the query phase, we have

$$\mathbb{P}\left(\text{accept} \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right) = \mathbb{P}\left(Q_1, ..., Q_m \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right),$$

so by independence of the choices of the s_0's and since the process is always the same,

$$\mathbb{P}\left(\text{accept} \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right) = \prod_{j=1}^{m} \mathbb{P}\left(Q_j \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right) = \mathbb{P}\left(Q_1 \,\middle|\, \bigcap_{i=0}^{r-1} \overline{E_i}\right)^m.$$

So we only have to do the proof for one step of the query phase. Futhermore, we now suppose that we have $\bigwedge_{i=0}^{r-1} \overline{E_i}$.

Consider the graph $G = (V, E)$ where $V = L_0 \sqcup ... \sqcup L_r$ and $(s, s') \in E$ iff there exists i such that $s \in L_{i+1}$, $s' \in L_i$ and $s = s'^2$. This graph is a tree with root the only element of L_r and leaves the elements of L_0. In these settings, the query phase consists in choosing a random leaf s_0 and verifying each step up to s_r.

Consider the following coloration c of the graph.

$$c(s) = \begin{cases} \text{green} & \text{if } s \in L_0 \\ \text{green} & \text{if } f_{i+1}(s) = \textbf{Fold}[f_i, \alpha_i](s) \\ \text{red} & \text{otherwise.} \end{cases}$$

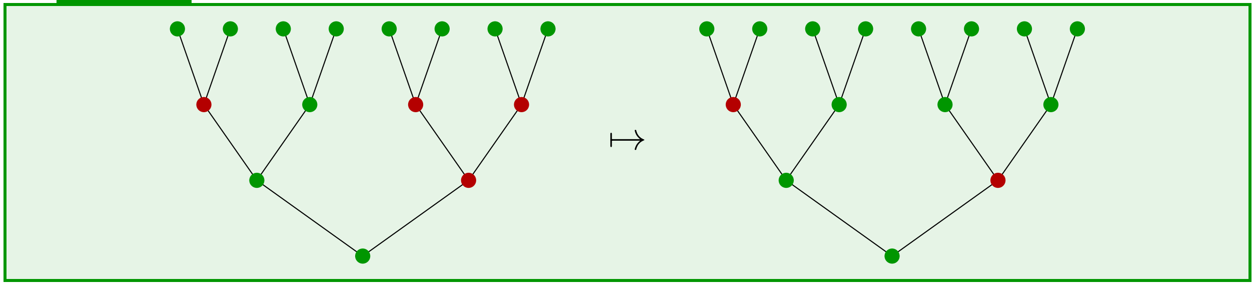
The Verifier accepts iff there is no red vertex in the branch from the chosen s_0 to s_r.

We are going to change the functions f_i into some f'_i in order to simplify the computation of the probability, where $\mathbb{P}(Q_1) \leqslant \mathbb{P}(Q'_1)$ with Q'_1 the event that the Verifier doesn't reject in the query phase with the functions f'_i.

In any branch $s_0...s_r$, starting from the root, when the first red node s_i is encountered, make all the s_j for $j < i$ correct, i.e. with $i = \max\{j \in [\![1, r]\!] \mid c(s_j) = \text{red}\}$,

$$f'_{j+1}(s_{j+1}) = \begin{cases} \textbf{Fold}[f_j, \alpha_j](s_j) & \text{if } j+1 < i \\ f_{j+1}(s_{j+1}) & \text{if } j \geqslant i. \end{cases}$$

Example



This way, with $\beta_i = \frac{|c'^{-1}(\text{red})|}{|L_i|}$ the proportion of red nodes in L_i,

- $\mathbb{P}(Q_1) \leqslant \mathbb{P}(Q'_1)$ because every branch of G that is accepting according to the f_i is accepting according to the f'_i
- $\Delta(f_{i+1}, \textbf{Fold}[f_i, \alpha_i]) \leqslant \beta_{i+1}$
- with the same notations, the new coloration c' of G satisfies $c'(s_j) = \begin{cases} \text{green} & \text{if } j < i \\ c(s_j) & \text{if } j \geqslant i. \end{cases}$

With this construction, in a branch with (at least) a red vertex according to c, there is exactly one red vertex according to c', and in a branch with no red vertex according to c, there is still no red vertex according to c'.

Therefore the event that the Verifier rejects is the disjoint union of the events C_i that the Verifier finds a vertex in L_i that is red according to c'. So we have

$$\begin{aligned} \mathbb{P}(Q_1) &= 1 - \mathbb{P}\left(\bigsqcup_{i=0}^{r} C_i\right) \\ &= 1 - \sum_{i=0}^{r} \mathbb{P}(C_i) \\ &= 1 - \sum_{i=0}^{r} \beta_i. \end{aligned}$$

We now have to bound the β_i. We split in two cases, if $\delta_{i+1} \leqslant \delta_i - \varepsilon$ or not.

If $\delta_{i+1} > \delta_i - \varepsilon$ then $\beta_{i+1} \geqslant 0 \geqslant \delta_i - \delta_{i+1} - \varepsilon$.

If $\delta_{i+1} \leqslant \delta_i - \varepsilon$ then $\delta_{i+1} < \delta_i \leqslant \delta'$ so $\delta_{i+1} = \Delta(f_{i+1}, V_{i+1})$. So by the triangle inequality,

$$\delta_{i+1} = \Delta(f_{i+1}, V_{i+1}) \geqslant \underbrace{\Delta(\textbf{Fold}[f_i, \alpha_i], V_{i+1})}_{\geqslant \delta_i - \varepsilon} - \underbrace{\Delta(f_{i+1}, \textbf{Fold}[f_i, \alpha_i])}_{\leqslant \beta_{i+1}}.$$

So in both cases, $\beta_{i+1} \geqslant \delta_i - \delta_{i+1} - \varepsilon$. Therefore,

$$\begin{aligned} \sum_{i=0}^{r-1} \beta_i &\geqslant \sum_{i=0}^{r-1} (\delta_i - \delta_{i+1} - \varepsilon) \\ &= \delta_0 - \delta_r - r\varepsilon \\ &= \delta_0 - r\varepsilon. \end{aligned}$$

Thus

$$\mathbb{P}(\text{accept}) \leqslant \frac{r}{\varepsilon^3 |\mathbb{F}|} + (1 - \min(\delta, J_\varepsilon(J_\varepsilon(\Delta(V_0)))) + r\varepsilon)^m.$$

A.1.2 Improved proofs from [BKS18]

A.1.2.1 Detailled proof of Theorem 3.5

For $\alpha \in A$, we have that $\Delta(u^*, v^\alpha - \alpha u) < \delta - \varepsilon$.

Consider the graph $G = (A, E)$ where

$$E = \{(\alpha, \alpha') \mid \Delta(v^\alpha - \alpha u, v^{\alpha'} - \alpha' u) < \gamma\},$$

and $\mathrm{co}G = (A, A^2 \setminus E)$ its complement graph.

Let $S_A \subseteq A$ be a clique of $\mathrm{co}G$ and

$$S = \{v^\alpha - \alpha u \mid \alpha \in S_A\}.$$

The function $\alpha \mapsto v^\alpha - \alpha u$ is injective, otherwise there exists α, α' such that $v^\alpha - \alpha u = v^{\alpha'} - \alpha' u$, so $u \in V$. Therefore $|S| = |S_A|$.

Since S_A is a clique, we have $\Delta(S) \geqslant \gamma$, so by hypothesis $|B(u^*, \gamma^*) \cap S| \leqslant \mu^*$. But since $\Delta(u^*, v^\alpha - \alpha u) < \delta - \varepsilon \leqslant \gamma^*$ by definition of v^α, we get that $B(u^*, \gamma^*) \cap S = S$. So $|S_A| \leqslant \mu^*$. Therefore $\mathrm{co}G$ doesn't have any clique of size $> \mu^*$.

So by Turán's theorem, $\mathrm{co}G$ has at most $(1 - \frac{1}{\mu^* - 1})\frac{|A|(|A|-1)}{2}$ edges, so at most $(1 - \frac{1}{\mu^*})\frac{|A|(|A|-1)}{2}$ edges, so G has at least $\frac{1}{\mu^*}\frac{|A|(|A|-1)}{2}$ edges.

$\sum_{\alpha \in A} \deg_G(\alpha) = \frac{1}{\mu^*}|A|(|A| - 1)$ so there exists $\alpha_0 \in A$ such that $\deg_G(\alpha_0) \geqslant \frac{1}{\mu^*}|A| - 1$. Let

$$B = \{\alpha \in A \mid (\alpha_0, \alpha) \in E\}.$$

Then $|B| \geqslant \frac{1}{\mu^*}|A| - 1$ and for all $\alpha \in B$,

$$\Delta(v^{\alpha_0} - \alpha_0 u, v^\alpha - \alpha u) = \Delta(u, \frac{1}{\alpha - \alpha_0}(v^\alpha - v^{\alpha_0})) < \gamma.$$

Let $C_v = \left\{\alpha \in B \mid v = \frac{1}{\alpha_0 - \alpha}(v^\alpha - v^{\alpha_0})\right\}$ and $V_u = V \cap B(u, \gamma)$. Then B is the disjoint union $\bigcup_{v \in V_u} C_v$, so $|B| = \sum_{v \in V_u} |C_v|$. Therefore there exists v such that $|C_v| \geqslant \frac{1}{|V_u|}|B|$.

By hypothesis on u, we have that $|B(u, \gamma) \cap V| \leqslant \mu$, so $|C_v| \geqslant \frac{1}{\mu}|B|$.

For all $\alpha \in C_v$,

$$v = \frac{1}{\alpha - \alpha_0}(v^\alpha - v^{\alpha_0}),$$

so with

$$v^* = v^{\alpha_0} - \alpha_0 v,$$

we have that for all $\alpha \in C_v$, $v^\alpha = v^* + \alpha v$.

So by definition of v^α, for all $\alpha \in C_v$,

$$\Delta(u^* - v^*, \alpha(v - u)) < \delta - \varepsilon.$$

Let $\overline{T} = \{i \in [\![1, n]\!] \mid (u_i^*, u_i) \neq (v_i^*, v_i)\}$. Let $D_\alpha = \{i \in \overline{T} \mid u_i^* - v_i^* = \alpha(v_i - u_i)\}$.

For $i \in \overline{T}$ there can be at most one $\alpha \in C_v$ such that $u_i^* - v_i^* = \alpha(v_i - u_i)$ ($\frac{u_i^* - v_i^*}{v_i - u_i}$ if $v_i \neq u_i$, and none if $v_i = u_i$ and $v_i^* \neq u_i^*$). I.e. for $i \in \overline{T}$ there is at most one $\alpha \in C_v$ such that $i \in D_\alpha$. So we have $\sum_{\alpha \in C_v} |D_\alpha| \leqslant |\overline{T}|$ so there exists α_1 such that $|D_{\alpha_1}| \leqslant \frac{|\overline{T}|}{|C_v|} \leqslant \frac{n}{|C_v|}$.

If $i \in \overline{T}$, and if $i \notin D_{\alpha_1}$, we have that $u_i^* - v_i^* \neq \alpha_1(v_i - u_i)$. Thus with $T = [\![1, n]\!] \setminus \overline{T}$,

$$\delta - \varepsilon > \Delta(u^* - v^*, \alpha_1(v - u)) \geqslant \underbrace{\underbrace{1 - \frac{|T|}{n}}_{i \notin T} - \frac{|D_{\alpha_1}|}{n}}_{i \notin D_{\alpha_1}} \geqslant 1 - \frac{|T|}{n} - \frac{1}{|C_v|}.$$

Since $|C_v| \geqslant \frac{1}{\mu}|B| \geqslant \frac{1}{\mu}\left(\frac{1}{\mu^*}|A| - 1\right) \geqslant \frac{1}{\varepsilon}$,

$$\frac{|T|}{n} > 1 - \delta + \varepsilon - \frac{1}{|C_v|} \geqslant 1 - \delta.$$

A.1.2.2 Detailled proof of Lemma 3.8

Suppose by a way of contradiction that $|A| > \mu$. Then there exists $T \subseteq L_{i+1}$ and $g_1, g_2 \in V_{i+1}$ such that $\frac{|T|}{|L_{i+1}|} > 1 - \delta$, $f_{0|T} = g_{1|T}$ and $f_{1|T} = g_{2|T}$.

Let $Q_1(Y)$ and $Q_2(Y)$ be the polynomials interpolating g_1 and g_2 respectively. Then $\deg(Q_1), \deg(Q_2) < 2^{k-i-1}$ by definition of V_{i+1}.

Let $Q(X,Y) = Q_1(Y) + XQ_2(Y)$. Then $\deg_Y(Q) \leqslant 2^{k-i-1} - 1$.

Let $R(X) = Q(X, X^2) = Q_1(X^2) + XQ_2(X^2)$. Then

$$\deg(R) \leqslant 2\deg_Y(Q) + 1 \leqslant 2^{k-i}) - 1 < 2^{k-i}.$$

So $R \in V_i$ and thus $\Delta(f, V_i) \leqslant 1 - \frac{|T|}{|L_{i+1}|} < \delta$, which contradicts our assumption. Therefore $|A| \leqslant \mu$.

A.2 DEEP-FRI commit soundness proof

Lemma A.1

Suppose that $\underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \eta$. Then

$$\underset{\alpha}{\mathbb{P}}\left(\underset{z}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \frac{\eta}{2}\right) \geqslant \frac{\eta}{2}.$$

Proof.
Let A_α be the event $\underset{z}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \frac{\eta}{2}$.
Suppose that $\underset{\alpha}{\mathbb{P}}(A_\alpha) < \frac{\eta}{2}$.
Then

$$\begin{aligned}
\underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z]) &= \underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z] \mid A_\alpha)\underset{\alpha,z}{\mathbb{P}}(A_\alpha) + \underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z] \mid \overline{A_\alpha})\underset{\alpha,z}{\mathbb{P}}(\overline{A_\alpha}) \\
&\leqslant \underbrace{\underset{\alpha,z}{\mathbb{P}}(A_\alpha)}_{<\frac{\eta}{2}} + \underbrace{\underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z] \mid \overline{A_\alpha})}_{<\frac{\eta}{2}} && \text{by Fubini} \\
&< \eta.
\end{aligned}$$

So if $\underset{\alpha,z}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \eta$, then $\underset{\alpha}{\mathbb{P}}(A_\alpha) \geqslant \frac{\eta}{2}$.

Notation A.2

Let V be a linear code $[n,k,d]_q$ of generating matrix $G \in \mathbb{F}^{k\times n}$. Let $D \subseteq \mathbb{F}^k$ be the set of columns of G. A codeword $v : D \to \mathbb{F}$ is the evaluation of a linear form ℓ_v.
Let $S \subseteq \mathbb{F}^k$ be a σ-robust set (every subset of S of size σ contains a basis of $\mathbb{F}^k$).

- For $a, b \in \mathbb{F}^k$, let $\langle a,b\rangle = \sum_{i=1}^k a_i b_i$.
- For $z \in S$ and $b \in \mathbb{F}$, let $V^{z,b} = \{v \in V \mid v = G\cdot \ell_v \text{ and } \langle \ell_v, z\rangle = b\}$.

Notation A.3

For $w \in [0,1]^n$ a weight and $u, v : \mathbb{F}_q^n$, we note the w-agreement between u and v by

$$\text{agree}_w(u,v) = \frac{1}{n}\sum_{\substack{1\leqslant i\leqslant n \\ u_i = v_i}} w_i.$$

And for $V \subseteq \mathbb{F}_q^n$,

$$\text{agree}_w(u,V) = \max_{v\in V} \text{agree}_w(u,v).$$

Proposition A.4 DEEP method for general linear codes with probability on the list bound (inspired from [BGKS20])

Let

- V be a linear code $[n,k,d]_q$ of generating matrix $G \in \mathbb{F}_q^{k\times n}$,
- $D \subseteq \mathbb{F}_q^k$ be the set of columns of G,
- $S \subseteq \mathbb{F}_q^k$ be a σ-robust set,
- $w \in [0,1]^n$,
- $u^*, u \in \mathbb{F}_q^n$, $\delta > 0$, $\mu \in \mathbb{N}^*$, and $0 < \varepsilon \leqslant \frac{1}{3}$.

Denote $E = \{\alpha \in \mathbb{F}_q \mid |B(u^* + \alpha u, \delta) \cap V| \leqslant \mu\}$ and $\eta = \max\left(2\mu\left(\frac{\sigma}{|S|} + \varepsilon\right)^{1/3}, \frac{4}{\varepsilon^2 q}\right)$.
Suppose that $\underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}(\alpha \in E) \geqslant p$ and that

$$\underset{\alpha\in\mathbb{F}_q, z\in S}{\mathbb{P}}(\mathrm{agree}_w(u^*+\alpha u, V^{z,B_z(\alpha)}) > 1-\delta) \geqslant p\eta + (1-p). \qquad (11)$$

Then there exists $T \subseteq [\![1,n]\!]$ such that $u^*_{|T} \in V_{|T}$, $u_{|T} \in V_{|T}$ and $\sum_{i\in T} w_i > (1-\delta-\varepsilon)n$.

Proof.
Let $D_{\alpha,z}$ be the event $(\mathrm{agree}_w(u^*+\alpha u, V^{z,B_z(\alpha)}) > 1-\delta$. By the total probabilities,

$$\underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(D_{\alpha,z}) = \underset{\alpha\in E,z\in S}{\mathbb{P}}(D_{\alpha,z}) \underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(\alpha\in E) + \underset{\alpha\in\overline{E},z\in S}{\mathbb{P}}(D_{\alpha,z}) \underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(\overline{\alpha\in E})$$

So by rearanging,

$$\begin{aligned}
\underset{\alpha\in E,z\in S}{\mathbb{P}}(D_{\alpha,z}) &= \frac{1}{\underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}(\alpha\in E)}\left(\underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(D_{\alpha,z}) - \left(1 - \underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}(\alpha\in E)\right)\underset{\alpha\in\overline{E},z\in S}{\mathbb{P}}(D_{\alpha,z})\right) \\
&\geqslant \frac{1}{p}\left(\underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(D_{\alpha,z}) - (1-p)\underset{\alpha\in\overline{E},z\in S}{\mathbb{P}}(D_{\alpha,z})\right) && \text{by hypothesis on } p \\
&\geqslant \frac{1}{p}\left(\underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(D_{\alpha,z}) - (1-p)\right) \\
&\geqslant \frac{1}{p}(p\eta + (1-p) - (1-p)) && \text{by hypothesis on } \underset{\alpha\in\mathbb{F}_q,z\in S}{\mathbb{P}}(D_{\alpha,z}) \\
&= \eta.
\end{aligned}$$

So $\underset{\alpha\in E,z\in S}{\mathbb{P}}(D_{\alpha,z}) \geqslant \eta$.
Let $u_\alpha = u^* + \alpha u$. Let $\mathcal{E}[\alpha,z]$ denote the event "$\exists v \in B(u_\alpha,\delta)\cap V, \langle v,z\rangle = B_z(\alpha)$ and $\mathrm{agree}_w(u_\alpha, v) > 1-\delta$". Then by assumption, $\underset{\alpha\in E,z\in S}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \eta$.

Thus, by Lemma A.1, $\underset{\alpha\in E}{\mathbb{P}}\left(\underset{z\in S}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \frac{\eta}{2}\right) \geqslant \frac{\eta}{2}$. Then with $A = \{\alpha\in E \mid \underset{z\in S}{\mathbb{P}}(\mathcal{E}[\alpha,z]) \geqslant \frac{\eta}{2}\}$, we have $|A| \geqslant \frac{\eta q}{2}$. For $\alpha\in E$, take

$$v_\alpha = \underset{v\in B(u_\alpha,\delta)\cap V}{\operatorname{argmax}} \underset{z\in S}{\mathbb{P}}(\langle v,z\rangle = B_z(\alpha)).$$

Let $S_\alpha = \{z\in S \mid \langle v_\alpha, z\rangle = B_z(\alpha)\}$ and $\mu_\alpha = \frac{|S_\alpha|}{|S|}$.
For $\alpha\in A$, there is a subset $S' \subseteq S$ such that $|S'| = \frac{\eta|S|}{2}$ and for all $z\in S'$, $\mathcal{E}[\alpha,z]$ is satisfied. This can be seen as a function $f_\alpha : S' \to B(u_\alpha,\delta)\cap V$ and S' is the disjoint union $\bigcup_{v\in V\cap B(u_\alpha,\delta)} f_\alpha^{-1}(\{v\})$, so $\sum_{v\in V\cap B(u_\alpha,\delta)} |f_\alpha^{-1}(\{v\})| = |S'| = \frac{\eta|S|}{2}$. Therefore there exists $v\in V\cap B(u_\alpha,\delta)$ such that $|f_\alpha^{-1}(\{v\})| \geqslant \frac{\eta|S|}{2|V\cap B(u_\alpha,\delta)|}$. And by definition, $\mu_\alpha = \mathbb{P}(\langle v_\alpha,z\rangle = B_z(\alpha)) \geqslant \mathbb{P}(\langle v,z\rangle = B_z(\alpha)) \geqslant |f_\alpha^{-1}(\{v\})|$. Furthermore, for $\alpha\in A\subseteq E$, we have $|B(u_\alpha,\delta)\cap V| \leqslant \mu$, so $\mu_\alpha \geqslant \frac{\eta}{2\mu}$.
For α,β,γ taken independently uniformly at random in A, we have

$$\underset{\alpha,\beta,\gamma\in A}{\mathbb{E}}\left(\frac{|S_\alpha\cap S_\beta\cap S_\gamma|}{|S|}\right) = \underset{z\in S,\alpha,\beta,\gamma\in A}{\mathbb{E}}(\mathbb{1}_{z\in S_\alpha\cap S_\beta\cap S_\gamma})$$

$$\begin{aligned}
&= \mathop{\mathbb{E}}_{z\in S}\Big(\mathop{\mathbb{E}}_{\alpha\in A}(\mathbb{1}_{z\in S_\alpha})^3\Big) && \text{by independence}\\
&\geqslant \mathop{\mathbb{E}}_{z\in S,\alpha\in A}(\mathbb{1}_{z\in S_\alpha})^3 && \text{by Jensen}\\
&= \mu_\alpha^3\\
&\geqslant \left(\frac{\eta}{2\mu}\right)^3 && \text{since } \alpha\in A\\
&\geqslant \frac{\sigma}{|S|}+\varepsilon. && \text{by definition of } \eta
\end{aligned}$$

Therefore

$$\begin{aligned}
\sigma+|S|\varepsilon &\leqslant \mathop{\mathbb{E}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|)\\
&= \sum_{k=1}^{|S|}\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|\geqslant k)\\
&= \sum_{k=1}^{\sigma}\underbrace{\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|\geqslant k)}_{\leqslant 1}+\sum_{k=\sigma+1}^{|S|}\underbrace{\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|\geqslant k)}_{\leqslant \mathbb{P}(|S_\alpha\cap S_\beta\cap S_\gamma|\geqslant\sigma+1)}\\
&\leqslant \sigma+(|S|-\sigma)\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma)\\
&\leqslant \sigma+|S|\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma).
\end{aligned}$$

So $\mathop{\mathbb{P}}_{\alpha,\beta,\gamma\in A}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma)\geqslant\varepsilon$.

$$\begin{aligned}
\mathbb{P}(\alpha,\beta,\gamma \text{ not all distinct}) &< \mathbb{P}(|\{\alpha,\beta,\gamma\}|=2)\\
&= \mathbb{P}(\alpha=\beta)+\mathbb{P}(|\{\alpha,\beta,\gamma\}|=2\mid\alpha\neq\beta)\\
&= \mathbb{P}(\alpha=\beta)+\mathbb{P}(\alpha=\gamma)+\mathbb{P}(|\{\alpha,\beta,\gamma\}|=2\mid\alpha\neq\beta,\alpha\neq\gamma)\\
&= \mathbb{P}(\alpha=\beta)+\mathbb{P}(\alpha=\gamma)+\mathbb{P}(\beta=\gamma)\\
&= \frac{3}{|A|}.
\end{aligned}$$

Since $\varepsilon\leqslant\frac{1}{3}$, we have $|A|\geqslant\frac{\eta q}{2}\geqslant\frac{2}{\varepsilon^2}\geqslant\frac{6}{\varepsilon}$, so $\mathbb{P}(\alpha,\beta,\gamma \text{ not all distinct})\leqslant\frac{\varepsilon}{2}$.
Thus,

$$\begin{aligned}
&\mathbb{P}(|\{\alpha,\beta,\gamma\}|=3 \text{ and } |S_\alpha\cap S_\beta\cap S_\gamma|>\sigma)\\
&=\mathbb{P}(|\{\alpha,\beta,\gamma\}|=3)\mathbb{P}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma\mid|\{\alpha,\beta,\gamma\}|=3)\\
&=\mathbb{P}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma)-\mathbb{P}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma\mid|\{\alpha,\beta,\gamma\}|<3)\mathbb{P}(|\{\alpha,\beta,\gamma\}|<3)\\
&\hspace{20em}\text{by the total probabilities}\\
&\geqslant\mathbb{P}(|S_\alpha\cap S_\beta\cap S_\gamma|>\sigma)-\mathbb{P}(|\{\alpha,\beta,\gamma\}|<3)\\
&\geqslant\varepsilon/2.
\end{aligned}$$

So there exists distinct $\alpha_0,\beta_0\in A$ such that $\mathop{\mathbb{P}}_{\gamma\in A}(|S_{\alpha_0}\cap S_{\beta_0}\cap S_\gamma|>\sigma)\geqslant\varepsilon/2>0$.
Let $B=\{\gamma\in A\mid |S_{\alpha_0}\cap S_{\beta_0}\cap S_\gamma|>\sigma\}$. Then $|B|\geqslant|A|\frac{\varepsilon}{2}\geqslant\frac{1}{\varepsilon}$.
Consider $\gamma\in B$ and let $\tilde{S}_\gamma=S_{\alpha_0}\cap S_{\beta_0}\cap S_\gamma$.
We extend u^* and u to S by for all $z\in S\setminus[\![1,n]\!]$, $u^*(z)=B_z(0)$, $u(z)=B_z(1)$, and for all $\alpha\in E$, $u_\alpha(z)=u^*(z)+\alpha u(z)$.
(α_0,u_{α_0}), (β_0,u_{β_0}) and (γ,u_γ) are collinear, so $(\alpha_0,u_{\alpha_0|\tilde{S}})$, $(\beta_0,u_{\beta_0|\tilde{S}})$ and $(\gamma,u_{\gamma|\tilde{S}})$ are collinear. Since V is systematic, we define $v_\gamma(z)=\langle v_{\gamma|[\![1,\sigma]\!]},z\rangle$ to extend v_γ to $\tilde{S}$. Thus by definition of $\tilde{S}$, $(\alpha_0,v_{\alpha_0|\tilde{S}})$, $(\beta_0,v_{\beta_0|\tilde{S}})$ and $(\gamma,v_{\gamma|\tilde{S}})$ are collinear. Furthermore, S is σ-robust and $|\tilde{S}|>\sigma$ so v_γ is uniquely determined by $v_{\gamma|\tilde{S}}$, and so (α_0,v_{α_0}), (β_0,v_{β_0}) and (γ,v_γ) are collinear.
We write the line $(\alpha_0,v_{\alpha_0}),(\beta_0,v_{\beta_0})$ as $v^*+\gamma v$, so for all $\gamma\in B$, $v_\gamma=v^*+\gamma v$. Let

$$T=\{i\in[\![1,n]\!]\mid(u^*(i),u(i))=(v^*(i),v(i))\}$$

and $\overline{T} = [\![1, n]\!] \setminus T$. For $\gamma \in E$ and $i \in [\![1, n]\!]$,

$$u_\gamma(i) - v_\gamma(i) = (u^*(i) - v^*(i)) + \gamma(u(i) - v(i))$$

so for $i \in \overline{T}$ there exists at most one value $\gamma \in B$ such that $u_\gamma(i) = v_\gamma(i)$.
Let $B_\gamma = \{i \in \overline{T} \mid u_\gamma(i) = v_\gamma(i)\}$. Then $\sum_{\gamma \in B} |B_\gamma| \leqslant |\overline{T}|$, so there exists $\gamma_0 \in A$ such that $|B_{\gamma_0}| \leqslant \frac{\overline{T}}{|B|}$. Thus

$$\begin{aligned}
1 - \delta &< \mathop{\mathbb{E}}_{\gamma \in B}(\mathsf{agree}_w(u_\gamma, v_\gamma)) && \text{by definition of } v_\gamma \\
&= \frac{1}{|B|} \sum_{\gamma \in B} \mathsf{agree}_w(u_\gamma, v_\gamma) \\
&= \frac{1}{n|B|} \sum_{\gamma \in B} \sum_{i=1}^{n} w_i \mathbb{1}_{u_{\gamma,i} = v_{\gamma,i}} && \text{by definition of agree} \\
&= \frac{1}{n} \sum_{i=1}^{n} w_i \left(\frac{1}{|B|} \sum_{\gamma \in B} \mathbb{1}_{u_{\gamma,i} = v_{\gamma,i}} \right) \\
&\leqslant \frac{1}{n} \sum_{i \in T} w_i + \frac{1}{n} \sum_{i \in \overline{T}} w_i \frac{1}{|B|} \\
&\leqslant \frac{1}{n} \sum_{i \in T} w_i + \frac{|\overline{T}|}{n|B|} \\
&\leqslant \frac{1}{n} \sum_{i \in T} w_i + \varepsilon && \text{because } |B| \geqslant \tfrac{1}{\varepsilon}
\end{aligned}$$

So $\sum_{i \in T} w_i > (1 - \delta - \varepsilon)n$.

With this proven, the exact proof of Proposition 4.6 is the following.

Proof of Proposition 4.6
Take $S = \{(z^i)_{0 \leqslant i \leqslant k-1} \mid z \in \mathbb{F}_q\}$. Then S is $(n-k)$-robust and it is equivalent to take an element of S or an element of $\mathbb{F}_q$. For $v \in V$, we have that ℓ_v is the coefficients of an interpolating polynomial of v, so $\langle \ell_v, (z^i)_{0 \leqslant i \leqslant k-1} \rangle = \sum_{i=0}^{k} (\ell_v)_i z^i = \ell_v(z)$.

B DEEP-FII protocol

Definition B.1 DEEP-FII protocol

Prover wants to prove that $f_0 : \mathcal{D}_0 \to \mathbb{F}_q^\ell \in V_0$.

commit

$$\begin{array}{ccc}
\text{Prover} & & \text{Verifier} \\
 & \xleftarrow{\quad z_0 \quad} & z_0 \xleftarrow{\$} \mathbb{F}^\ell \\
B_{1,z_0} : \mathbb{F}^\ell \to \mathbb{F}^\ell & & \\
(B_{1,z_0}(X) = \textbf{Fold}[f_0, X](z_0)) & \xrightarrow[\quad B_{1,z_0} \quad]{} & \\
 & \xleftarrow{\quad \alpha_0 \quad} & \alpha_0 \xleftarrow{\$} \mathbb{F}^\ell \\
f_1 : \mathcal{D}_1 \to \mathbb{F}^\ell & & \\
(f_1(X) = \frac{\textbf{Fold}[f_0,\alpha_0](X) - B_{1,z_0}(\alpha_0)}{X - z_0}) & \xrightarrow[\quad f_1 \quad]{} & \\
 & & \vdots \\
\vdots & \vdots & \\
 & \xleftarrow{\quad z_{r-1} \quad} & z_{r-1} \xleftarrow{\$} \mathbb{F}^\ell \\
B_{r,z_{r-1}} : \mathbb{F}^\ell \to \mathbb{F}^\ell & & \\
(B_{r,z_{r-1}}(X) = \textbf{Fold}[f_{r-1}, X](z_{r-1})) & \xrightarrow[\quad f_r \quad]{} & \\
 & \xleftarrow{\quad \alpha_{r-1} \quad} & \alpha_{r-1} \xleftarrow{\$} \mathbb{F}^\ell \\
f_r : \mathcal{D}_r \to \mathbb{F} & & \\
(f_r(X) = \frac{\textbf{Fold}[f_{r-1},\alpha_{r-1}](X) - B_{r,z_{r-1}}(\alpha_{r-1})}{X - z_{r-1}}) & \xrightarrow[\quad f_r \quad]{} &
\end{array}$$

query

Verifier chooses $s_0 \xleftarrow{\$} \mathcal{D}_0$, defines $s_{i+1} := s_i^2$ and accepts if

$$\left.\begin{array}{c}
\textbf{Fold}[f_0, \alpha_0](s_1) = (s_{i+1} - z_i) f_{i+1}(s_{i+1}) + B_{i+1,z_i}(\alpha_i) \\
\vdots \\
\textbf{Fold}[f_{r-1}, \alpha_{r-1}](s_r) = (s_r - z_{r-1}) f_r(s_r) + B_{r,z_{r-1}}(\alpha_{r-1}).
\end{array}\right) \text{repeat}$$

Note that the $B_{i,z_{i-1}}$ are not really multivariate functions, they are a succession of ℓ univariate functions.

Index of definitions

Index of results