Study of Arithmetization Methods for STARKs

Tiago Martins
tiago.martins@threesigma.xyz

João Farinha
joao.farinha@threesigma.xyz

Three Sigma - Academic Research

May 23, 2023

Abstract

This technical paper explores two solutions for arithmetization of computational integrity statements in STARKs, namely the algebraic intermediate representation, AIR, and is preprocessed variant, PAIR. The work then focuses on their soundness implications for Reed-Solomon proximity testing. It proceeds by presenting a comparative study of these methods, providing their theoretical foundations and deriving the degree bounds for low-degree proximity testing. The study shows that using PAIR increases the degree bound for Reed-Solomon proximity testing, which affects its soundness and complexity. However, the possibility of reducing the degree bound with multiple selector columns is also explored, namely by following an approach based on the decomposition of the selector values. Focusing on performance optimization, the work proceeds by qualitatively comparing computational demands of the components of both arithmetization methods, particularly their impact on the low-degree extensions. The paper concludes that, while PAIR might simplify constraint enforcement, it can be easily translated to AIR, and system testing with benchmarks is necessary to determine the application-specific superiority of either method. This work should provide insight into the strengths and limitations of each method, helping researchers and practitioners in the field of STARKs make informed design choices.
Key words. STARKs, Arithmetization, AIR, Preprocessed AIR, RPT, Degree bound, Selector columns.

1 Introduction

As the volume of transactions in blockchain technology continues to grow, scalability and performance optimization have become critical concerns. STARKs, which stands for Scalable Transparent Arguments of Knowledge, offer a promising solution for achieving scalability by allowing for efficient verification of complex computations on large amounts of data. However, to fully realize the benefits of STARKs, it is essential to consider the security implications and computational requirements of the underlying protocols, such as to promote soundness and optimizing performance for adequate scalability.

This technical work delves into two solutions for polynomial arithmetization of computational integrity statements, namely the AIR (Algebraic Intermediate Representation) and PAIR (Preprocessed AIR) protocols. The focus of the analysis is on the soundness implications of these two methods concerning low-degree proximity testing. The study examines how these solutions perform and promote tolerable soundness, which is a critical factor for smaller proof-sizes in STARKs. Additionally, this document briefly discusses the importance of optimizing performance for these methods given their varying computational requirements.

In this context, it is essential to demonstrate with high probability that a given function is close to a low-degree polynomial, which is typically achieved using Reed-Solomon arguments of proximity. Interactive Oracle Proofs of Proximity (IOPP) such as FRI (Fast Reed-Solomon IOPP) have been developed for this purpose, but exploring them is beyond the scope of this paper—for detailed descriptions, see [14, 3]. However, the establishment of tight degree bounds for Reed-Solomon proximity testing of the AIR and PAIR polynomials is critical in assuring the security of STARKs.

This paper compares the tightness of the degree bounds of these two methods, after providing their theoretical foundations. It then briefly explores computational complexity and performance characteristics of AIR versus PAIR. Hence, this document aims to offer researchers and professionals in the field of STARKs valuable information to make knowledgeable design decisions.

1.1 Background Analysis of the Arithmetization Process

Arithmetization is a technique adapted to interactive proof systems [1, 16]. It consists in the reduction of computational problems to algebraic problems, involving "low-degree" polynomials over a finite field—i.e. the degree is significantly smaller than the field size [5]. The arithmetization process employed in STARKs is comprised by different stages of algebraic transformations, which are typically distinguished as follows:

(a) An Algebraic Intermediate Representation (**AIR**) is a representation of a computational integrity statement given by a public instance, shared between the verifier and the prover, and a private witness, known only by the prover [4]. In the AIR, Boolean circuits are represented using strongly "structured" arithmetic circuits [17] that employ field algebraic operations on polynomials, such that the truth values in the circuit's wires describe whether these polynomials evaluate to zero on a given input [1]. AIRs were previously known as constraint satisfaction problems (ACSPs) in prior works like [9, 2].

(b) Subsequently, the Algebraic Placement and Routing (**APR**) reduction transforms the AIR instance and witness into a set of Reed-Solomon membership problems, with perfect completeness, soundness and knowledge extraction. After the APR reduction, instead of verifying satisfiability of the AIR instance (with the arithmetic circuit), the verifier checks whether some functions are *members* of particular Reed-Solomon codes [4].

(c) Finally, the Algebraic Linking IOP (**ALI**) is the last step of arithmetization and consists in combining the APR's Reed-Solomon *membership* problems into a smaller number of Reed-Solomon *proximity* problems, using public randomness [5, 4]. The public randomness helps to ensure that the final Reed-Solomon proximity testing (**RPT**) problems are well-formed and avoid any biases or security weaknesses that could arise from private-coins. Canonically, the public-coins come from additional interaction steps (the I in IOP, Interactive Oracle Proof), but these may be substituted by practically equivalent non-interactive procedures [11, 7]. Additionally, the ALI hinders dishonest provers from passing proximity tests with high-degree codes that are very close to low-degree in Hamming distance [8], since all the tested codes are random combinations of the APR codes, which decreases the likelihood of said vulnerability. The ALI reduction has perfect completeness, and almost perfect soundness and knowledge extraction.

Note that this document defines (section 3) and employs AIRs in alignment with the definition in [21], being a well acknowledged implementation of constraint arithmetization in STARKs, specifically. Nevertheless, this work will deal with all three distinct stages, since a change in the way that the constraints are represented by polynomials forces the adjustment of all subsequent stages including the Reed-Solomon proximity testing procedure, after the arithmetization.

1.2 The Discovery of PAIR

The primary motivation for this document arose from the analysis of the source [12], whose underlying concept—Preprocessed AIR (**PAIR**)—is also explored in [18]. This arithmetization notion appeared to be quite promising within the context of an execution trace subject to disjoint AIR constraints.

Both articles detailed how one could simplify the application of two such constraints by combining them into a single larger constraint, subject to an extra data input: the binary selector column. The final constraint would take the form:

$$\text{Constraint}_0(x) \times \big(1 - \text{selector}(x)\big) + \text{Constraint}_1(x) \times \text{selector}(x) = 0\,,$$

where x represents an execution trace domain point—further explained in section 2.1. This construction would enforce Constraint_0 when the selector assumes the value 0, and likewise for Constraint_1. However, both sources abstained from expanding on this idea, or rigorously inspecting its benefits and drawbacks; this sparked our interest in formalizing and generalizing the concept to more than two disjoint constraints, and analytically examine how this method would fare in the later stages of a STARK, namely concerning the degree bounds of the resulting polynomials for the RPT.

Another interesting property of the PAIR is that it eliminates the need for interactions in the arithmetization, when applied to all constraints, since its inherent algebraic linking already merges multiple algebraic problems

[1] Polygon [17] for example, introduced for their zkEVM the variation *Extended Algebraic Intermediate Representation,* eAIR, as an extension of the typical AIR: by imposing the simplifications that the constraint polynomials only apply at most to two consecutive trace domain points, and all constraints must evaluate to zero over the entire evaluation domain (instead of a subset of it), this method allows to define more broader constraint types, that may not only be represented by a polynomial (*identity constraints*), but by a set of polynomials and integers (*non-identity constraints*).

into a single one. In other words, if the PAIR combines all constraints without requiring external public-coins, then the algebraic linking of the constraints is non-interactive and the ALI is absent. In this case, PAIR would be characterized by perfect soundness and knowledge extraction [4], as it would become a perfect reduction of computational integrity problems, without associated error[2]. However, keep in mind that using the ALI will help separate high and low degree polynomials, prior to Reed-Solomon proximity testing—i.e. if the codes from arithmetization are of high degree, then the ALI transforms them into new codes that are likely far from those of low-degree [8, Section 1.4]. In this regard, performing the ALI promotes soundness in the subsequent RPT procedures, even if it introduces a small soundness error in the arithmetization.

At this point, it is relevant to unambiguously distinguish the designation PAIR - *Preprocessed* AIR, as in articles [12, 18] and as it will be used henceforth - from the also studied *Permuted* AIR, defined in [4]. The latter is a form of memory-consistency check, applied to a permutation of the steps of the execution trace.

Polygon Miden, for instance, in their recently developed constraint description language, AirScript [19], makes use of preallocated trace columns to serve as selectors. These selectors are implemented as "periodic columns", and although the application of the selector column is not completely akin to the description in section 4, the fundamental mechanic remains: the extra columns of information appended to the trace specify the constraints enforced in each row. StarkWare also describes, in the ethSTARK documentation [21], the use of periodic columns to address cyclic constants that are appended to the trace, employing two periodic columns per trace column.

2 Execution Trace

The execution trace of a program is a register of successive states of computation of a particular program/machine. These are typically distinguished by clock cycles and are assigned values that specifically encode the relevant parameters. In accordance, the states may either be represented by a collection of state variables (several different values), or a single value unequivocally determined from the aggregation of the different state variables. Minding the discrete nature of computational operation, finite fields are typically used for state variable assignments—also resolving the concern of numeric overflow and underflow. As such, the analysis below ensues within this context.

2.1 Formalizing the Execution Trace

Let $\mathbb{F}$ be a finite field. The vector-valued trace function will be defined as $\mathbf{f} : \mathbb{F} \to \mathbb{F}^W$, where $W \in \mathbb{N}$. The execution trace domain $\mathbb{G}$ is a multiplicative subgroup of $\mathbb{F}^\times$ generated by an element g. The execution trace is defined as a tuple constructed from the evaluation of the restriction of $\mathbf{f}$ to $\mathbb{G}$. Expressly, the execution trace is the finite sequence $\left(\mathbf{f}(g^k)\right)_{k=0}^{|\mathbb{G}|-1}$, which constitutes a $|\mathbb{G}|$-tuple. To achieve zero-knowledge, the computational integrity statement only restricts part of the execution trace, leaving some evaluation points to be randomly or freely filled-in by the prover [4].

In this context, all $\mathbb{F}$ to $\mathbb{F}$ functions are polynomial functions and this can be shown using Lagrange interpolation (see chapter 05.04 of [15]). The trace function $\mathbf{f}$ should be uniquely defined from its restriction to $\mathbb{G}$, i.e. from the execution trace. Moreover, each of the components of $\mathbf{f}$ is an $\mathbb{F} \to \mathbb{F}$ function and may be generated from the polynomial interpolation of $|\mathbb{G}|$ data points, thus having polynomial degree[3] at most $|\mathbb{G}| - 1$. Therefore it is also natural to state that the degree of $\mathbf{f}$ satisfies $\deg(\mathbf{f}) \leq |\mathbb{G}| - 1$.

3 Algebraic Intermediate Representation (AIR)

In STARKs, an algebraic intermediate representation is a high-level description of arithmetic constraints for computations. The computational integrity statement is enforced via constraints which are expressed as polynomials composed over the execution trace. In this work, the AIR is defined based on previous AIR definitions, like [4, 21], and is stated using multiplicative groups.

When building a specific proof, the utilized AIR must be defined upon agreement between the verifier and the prover, implying that it is public. The corresponding constraints express certain conditions which, composed over the execution trace, hold if and only if there is computational integrity. Ultimately, an AIR is a collection of

(i) a large finite field $\mathbb{F}$,

(ii) a number $W \in \mathbb{N}$, specifying the dimension of the codomain of the trace function,

[2]This does not remove the error introduced in further stages: for instance, when executing IOPs for RPT, such as FRI.

[3]Recall that the degree of an univariate polynomial is the highest occurring exponent, disregarding those associated to null coefficients.

(iii) a multiplicative subgroup $\mathbb{G} < \mathbb{F}^\times$, used as the execution trace domain,

(iv) a generator g of $\mathbb{G}$, and

(v) a set of constraints, with index set $\mathcal{B} \subset \mathbb{F}$, where, for $i \in \mathcal{B}$, the ith constraint, $(\mathbb{G}_i, \mathbf{M}_i, C_i)$, is a tuple of

  (a) the ith constraint enforcement domain, the set $\mathbb{G}_i \subset \mathbb{G}$,
  (b) the ith mask, $\mathbf{M}_i$, which is a tuple of length L_i, and
  (c) the multivariate polynomial of the ith constraint, $C_i : \mathbb{F}^{L_i} \to \mathbb{F}$.

Moreover, the mapping $(x_0, \ldots, x_{L_i-1}) \mapsto C_i(x_0, \ldots, x_{L_i-1})$ is a multivariate polynomial, whose degree, $\deg(C_i)$, is defined as the degree of the univariate polynomial $x \mapsto C_i(x, \ldots, x)$, for each $i \in \mathcal{B}$.

In order to compute the constraints in a well determined and concise manner, let us define the ith mask $\mathbf{M}_i$ as the finite tuple that is used to evaluate the ith constraint. Expressly, the ith mask is the tuple:

$$\begin{aligned}&\mathbf{M}_i = \big((m_{i,j}, \mathbf{u}_{i,j})\big)_{j=0}^{L_i-1} \\ &\text{where} \quad m_{i,j} \in \mathbb{G} \quad \text{and} \quad \mathbf{u}_{i,j} \in \mathbb{F}^W \quad \text{for all} \quad j \in \{0, \ldots, L_i - 1\}\,.\end{aligned}$$

Regarding the jth argument of C_i, the scalar $m_{i,j}$ is the x offset for the evaluation of $\mathbf{f}$, and $\mathbf{u}_{i,j}$ is utilized to linearly transform the vector $\mathbf{f}(x\, m_{i,j})$ into a scalar—that is, it forms a linear combination of vector components. Subsequently, the ith mask may be used to express the ith constraint:

$$C_i\big(\mathbf{f}(x\, m_{i,0}) \cdot \mathbf{u}_{i,0}, \ldots, \mathbf{f}(x\, m_{i,L_i-1}) \cdot \mathbf{u}_{i,L_i-1}\big) = 0 \quad \text{for all} \quad x \in \mathbb{G}_i\,,$$

where $\cdot$ stands for the dot product. For further conciseness, define the mapping

$$x \mapsto \psi_i(x) = C_i\big(\mathbf{f}(x\, m_{i,0}) \cdot \mathbf{u}_{i,0}, \ldots, \mathbf{f}(x\, m_{i,L_i-1}) \cdot \mathbf{u}_{i,L_i-1}\big)\,.$$

Since the AIR expresses the computational integrity conditions, when evaluating ψ_i at the corresponding enforcement domain, the constraint will hold if and only if the result is zero. As such, the representation of the constraints (rules) for the execution trace function $\mathbf{f}$ are statements of the form: $\psi_i(x) = 0$ for all $x \in \mathbb{G}_i$.

Now, the tuple $(\mathbb{G}_i, \mathbf{M}_i, C_i)$ unambiguously defines the mathematical expression of the ith constraint. In short, the AIR assignment satisfies the algebraic intermediate representation if and only if

$$\psi_i(x) = 0 \quad \text{for all} \quad x \in \mathbb{G}_i\,, \quad \text{for all} \quad i \in \mathcal{B}\,.$$

Note that, if the function $\mathbf{f}$ is indeed a polynomial of degree $\deg(\mathbf{f}) \leq |\mathbb{G}| - 1$, then $\psi_i(x)$ is a polynomial of degree $\deg(\psi_i) \leq \deg(C_i) \times \big(|\mathbb{G}| - 1\big)$, as the degree of the composition of two non-constant polynomials over a field is the product of their degrees.

The prover is the only one that knows the function $\mathbf{f}$, thus they are the only ones that can independently evaluate ψ_i and determine its exact degree. However, the verifier is also aware of the definition of the AIR, hence they will be able to compute $\psi_i(x)$ if they have knowledge of the scalars $\mathbf{f}(x\, m_{i,j}) \cdot \mathbf{u}_{i,j}$ for all $j \in \{0, \ldots, L_i - 1\}$.

3.1 Algebraic Placement and Routing in AIR

In APR, the AIR instance and witness are directly mapped to a set of Reed-Solomon membership problems. To build this new framework, define the vanishing polynomial of a set $\mathcal{S}$ as the unique monic polynomial of degree $|\mathcal{S}|$ whose set of roots is precisely $\mathcal{S}$ (each root having multiplicity one). Thus, the vanishing polynomial of $\mathcal{S}$ is

$$x \mapsto V_{\mathcal{S}}(x) = \prod_{y \in \mathcal{S}} (x - y)\,. \tag{1}$$

If the function ψ_i has a root for every element of $\mathbb{G}_i$ (an honest prover situation), then the expression

$$x \mapsto Q_i(x) = \frac{\psi_i(x)}{V_{\mathbb{G}_i}(x)}$$

defines a polynomial Q_i of degree:

$$\deg(Q_i) = \deg(\psi_i) - |\mathbb{G}_i| \leq \deg(C_i) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}_i| = \mathfrak{d}_{Q_i}\,, \tag{2}$$

yielding a tight upper bound for the degree of Q_i, acknowledged by both the verifier and the prover. Introducing this new polynomial Q_i seems to add unnecessary complexity into the previous mathematical expression; however, Q_i is now suitable for Reed-Solomon membership testing, as follows.

(i) In the case of an honest prover, the numerator will have factors that cancel the vanishing polynomial in the denominator, since $\psi_i(x) = 0$ for any $x \in \mathbb{G}_i$, effectively resulting in a Q_i with degree at most $\mathfrak{d}_{Q_i}$.

(ii) For a dishonest prover, at least one of the constraints will fail somewhere in its enforcement domain, therefore, the denominator of that Q_i will not be cancelled, resulting in a rational function that is not a polynomial of degree at most $\mathfrak{d}_{Q_i}$ [4].

Thus, APR yields constraints that are practically equivalent to the corresponding original computational integrity constraints.

3.2 Algebraic Linking IOP in AIR

ALI, the last step of arithmetization, reduces the amount of Reed-Solomon proximity tests by building the composition polynomial. The overall idea is that public randomness is used to combine the Q_i polynomials into a single low-degree polynomial [4]. This induces a new verifier-prover interaction to exchange this public-coin, which may be made non-interactive using the Fiat-Shamir transform [11, 7].

ALI also removes some of the prover's influence on the proximity testing, by making it computationally unfeasible and probabilistically unlikely that the prover is able to manipulate the APR polynomials such that the final ALI polynomials lie within a desired outcome for the subsequent proximity testing—i.e. it hampers RPT satisfiability of high-degree codes that resemble those of low-degree, in the APR stage. To put it differently, when the input instance is unsatisfiable, the ALI generates RPT problem instances that are significantly far from the relevant Reed-Solomon codes [8].

The APR constraints are combined once the prover has committed to the trace low degree extension[4]. The degree of Q_i is bounded as $\deg(Q_i) \leq \mathfrak{d}_{Q_i}$, therefore, considering all constraints, the maximum degree bound is $\mathfrak{d}_{\text{CP}} = \max_{j\in\mathcal{B}}\{\mathfrak{d}_{Q_j}\}$. For the simplest ALI without degree-adjustment, for each Q_i constraint the verifier provides a random scalar α_i in the extension field[5] $\mathbb{K}$. In this manner, the composition polynomial becomes

$$x \mapsto \text{CP}(x) = \sum_{i\in\mathcal{B}} \alpha_i \, Q_i(x) \,,$$

and the Reed-Solomon proximity test must check whether $\deg(\text{CP}) \leq \mathfrak{d}_{\text{CP}}$ with high probability.

In a more sophisticated approach [section D.1 of 4], the Q_i constraints are instead multiplied by a degree-adjustment polynomial in order to ensure that all of the constraints are treated equally in the low-degree analysis. In this case, the verifier provides random polynomials [10] α_i, that are utilized to adjust the degree of Q_i [21]. Expressly, α_i is now a publicly-random polynomial over $\mathbb{K}$ such that

$$\deg(\alpha_i) = \mathfrak{d}_{\text{CP}} - \mathfrak{d}_{Q_i} \,,$$

which adjusts the degree-bound of the corresponding constraint as

$$\deg(\alpha_i Q_i) = \mathfrak{d}_{\text{CP}} - \mathfrak{d}_{Q_i} + \deg(Q_i) \leq \mathfrak{d}_{\text{CP}} \,. \tag{3}$$

In short, all the degree-adjusted constraints will have the same tight degree bound. However, the verifier should not provide polynomials α_i with roots in $\mathbb{G}_i$, expressly $\alpha_i(\mathbb{G}_i) \cap \{0\} = \emptyset$ for any $i \in \mathcal{B}$, since wrong choices of the random polynomial α_i will undermine the whole proof, as any additional common factors between the denominator and numerator of $\alpha_i Q_i$ make it impossible to conclude that those same factors were common to ψ_i and $V_{\mathbb{G}_i}$. In ethSTARK Version 1.1 for example, α_i is a polynomial with just the lowest and highest degree terms (all others are zero) [Section 3.6.1 of 21].

As a result, if the new function $\alpha_i Q_i$ is a polynomial of degree at most $\mathfrak{d}_{\text{CP}}$, then the original function Q_i has degree at most $\mathfrak{d}_{Q_i}$, as desired. The composition polynomial CP then sums all $\alpha_i Q_i$ polynomials into a unidimensional univariate polynomial:

$$\begin{aligned} &\text{CP} : \mathbb{K} \to \mathbb{K} \\ &\qquad x \mapsto \text{CP}(x) = \sum_{i\in\mathcal{B}} \alpha_i(x) \, Q_i(x) \,, \end{aligned}$$

[4]The trace low degree extension is the evaluation of $\mathbf{f}$ over a large domain, disjoint from $\mathbb{G}$ [21]. The former is commonly a non-trivial coset of an overgroup of $\mathbb{G}$. This will be further explained in section 7.

[5]The field $\mathbb{K}$ is an an algebraic extension of $\mathbb{F}$, i.e. all the elements of $\mathbb{K}$ are roots of polynomials over $\mathbb{F} \subset \mathbb{K}$. Moreover, selecting uniformly random elements from a larger field helps promote security in the protocol [21].

which should have degree $\deg(\mathrm{CP}) \leq \mathfrak{d}_{\mathrm{CP}}$, yielding a Reed-Solomon proximity test.

Succinctly, once the prover has committed to the trace low degree extension[4], the verifier provides random scalars (or random coefficients in the degree-adjustment case) for creating a random linear combination of the constraints, resulting in the composition polynomial. Instead of performing a low-degree test on each of the constraints, it suffices to test the composition polynomial. If the composition polynomial CP has degree at most $\mathfrak{d}_{\mathrm{CP}}$, then there is a high probability (over the choice of the random α_i polynomials) that the assignment satisfies the computational integrity statement. It is very unlikely that a random linear combination of high-degree polynomials yields a polynomial of lower degree. However, that may occur due to the small but non-null probability of the higher degree terms cancelling[6]. Nonetheless, if the final summation is of low-degree, then the same is likely about the individual polynomial summands. And naturally, if there is computational integrity, CP has degree at most $\mathfrak{d}_{\mathrm{CP}}$, accomplishing perfect completeness. Theorem 3.1 will now clarify which AIR constraints determine $\mathfrak{d}_{\mathrm{CP}}$.

Theorem 3.1. *The degree bound for the composition polynomial, $\mathfrak{d}_{\mathrm{CP}}$, will be determined by a Q_i rational polynomial whose C_i polynomial has maximal degree. Equivalently,*

$$\mathfrak{d}_{\mathrm{CP}} = \max_{j\in\mathcal{B}}\{\mathfrak{d}_{Q_j}\} = \mathfrak{d}_{Q_i}\,, \quad \textit{for some} \quad i \in \mathcal{B} \quad \textit{satisfying} \quad \deg(C_i) = \max_{j\in\mathcal{B}}\big\{\deg(C_j)\big\}\,.$$

Proof. Firstly, choose $i \in \mathcal{B}$ such that $\deg(C_i) = \max_{j\in\mathcal{B}}\{\deg(C_j)\}$. For any $k \in \mathcal{B}$ such that $\deg(C_k) < \deg(C_i)$ one has $\deg(C_k) \leq \deg(C_i) - 1$, since polynomial degrees are natural numbers. Furthermore, for all $j \in \mathcal{B}$, the enforcement domain satisfies $1 \leq |\mathbb{G}_j| \leq |\mathbb{G}|$. Consequently,

$$\begin{aligned}\mathfrak{d}_{\mathrm{CP}} \geq \mathfrak{d}_{Q_i} &= \deg(C_i) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}_i| \geq \deg(C_i) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}| = \\ &= \big[\deg(C_i) - 1\big] \times \big(|\mathbb{G}| - 1\big) - 1 \geq \deg(C_k) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}_k| = \mathfrak{d}_{Q_k}\,.\end{aligned}$$

Hence, for any $i, k \in \mathcal{B}$ such that $\deg(C_k) < \deg(C_i) = \max_{j\in\mathcal{B}}\{\deg(C_j)\}$ one has $\mathfrak{d}_{\mathrm{CP}} \geq \mathfrak{d}_{Q_i} \geq \mathfrak{d}_{Q_k}$, which implies the existence of $i \in \mathcal{B}$ such that $\mathfrak{d}_{Q_i} = \max_{j\in\mathcal{B}}\{\mathfrak{d}_{Q_j}\} = \mathfrak{d}_{\mathrm{CP}}$ and $\deg(C_i) = \max_{j\in\mathcal{B}}\{\deg(C_j)\}$. □

Table 1 summarizes the distribution of knowledge, among the proof intervenients, of the composition polynomial components. Naturally, the prover is aware of all information.

Table 1: Knowledge of CP components. The comparison applies to all $i \in \mathcal{B}$.

	$\mathbb{F}$	$\mathbb{G}$	$\mathbf{f}$	$\mathbb{G}_i$	$\mathbf{M}_i$	C_i	ψ_i	Q_i	$\mathfrak{d}_{Q_i}$	α_i	CP	$\mathfrak{d}_{\mathrm{CP}}$
Verifier (public)	yes	yes	**no**	yes	yes	yes	**no**	**no**	yes	yes	**no**	yes
Prover (private)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

The key takeaway is that, as expected, the execution trace itself is the sole concern of privacy; consequently, the full representation of the objects that derive from it is also concealed from the verifier, namely ψ_i, Q_i, and CP.

4 Preprocessed AIR (PAIR)

Preprocessed AIR is an algorithm that aims to simplify the construction of an AIR in the particular case of disjoint constraints: if two constraints are never meant to operate simultaneously (i.e. their enforcement domains are disjoint), then they can be combined into a single constraint. This is achieved by appending an extra column of data to the trace matrix (hence *preprocessed*), that will act as a selector for the constraint. The typical example would be the situation where specific state variables in the execution trace must be the sum of a set of state variables from the previous domain element, where others would be the product, as illustrated in [12] using a binary selector (for two disjoint constraints).

4.1 Generalization for $|\mathcal{B}|$ Constraints

It is possible to generalize the idea of the binary selector in order to combine multiple constraints. Choose $\mathcal{B}$ as the index set of the disjoint AIR constraints to be combined into a single "larger" constraint. The generalized combined constraint joins $|\mathcal{B}|$ disjoint AIR constraints, for which $\mathbb{G}_i \cap \mathbb{G}_j = \emptyset$ if $i \neq j$ where $i, j \in \mathcal{B}$. In this case,

[6] This is described in lemma 3.2 of [8], without degree-adjustment.

the selector function s will take values in the set $\mathcal{B}$ such that it selects just one of the possible $Q_{i\in\mathcal{B}}$ for activation, namely

$$s : \mathbb{G}_{\mathcal{B}} \to \mathcal{B} \quad \text{with} \quad \mathbb{G}_{\mathcal{B}} = \bigsqcup_{i\in\mathcal{B}} \mathbb{G}_i \,,$$

where $\mathbb{G}_{\mathcal{B}}$ is the enforcement domain of the combined constraint. Recall that, originally, the constraints indexed by $\mathcal{B}$ were of the form $\psi_i(x) = 0$ for all $x \in \mathbb{G}_i$ for all $i \in \mathcal{B}$. To suppress the need for the constraint enforcement domains, define the selector function such that it yields

$$s(x) = i \quad \text{for all} \quad x \in \mathbb{G}_i \quad \text{for all} \quad i \in \mathcal{B}.$$

Moreover, this may be accomplished by defining the selector function (of lowest polynomial degree) as

$$s(x) = \sum_{i\in\mathcal{B}} i \sum_{z\in\mathbb{G}_i} l(x \mid z) \quad \text{such that} \quad s(x) = i \quad \text{for all} \quad x \in \mathbb{G}_i \quad \text{for all} \quad i \in \mathcal{B}\,, \tag{4}$$

where the Lagrange basis for polynomial interpolation in $\mathbb{G}_{\mathcal{B}}$ is given by

$$l(x \mid z) = \frac{V_{\mathbb{G}_{\mathcal{B}}\setminus\{z\}}(x)}{V_{\mathbb{G}_{\mathcal{B}}\setminus\{z\}}(z)} \quad \text{which yields} \quad l(x \mid z) = \begin{cases} 1\,, & x = z \\ 0\,, & x \neq z \end{cases} \quad \text{for} \quad x, z \in \mathbb{G}_{\mathcal{B}}\,,$$

recalling that $V_{\mathbb{G}_{\mathcal{B}}\setminus\{z\}}$ is defined in accordance with equation (1). Also, the polynomials of the selector function may be computed with a fast Fourier transform algorithm (the same as $\mathbf{f}$) (see chapter 9 of [22]).

With the selector function unambiguously defined, by constructing the polynomial $\psi_{\mathcal{B}}$ the combined constraint is expressed as

$$\psi_{\mathcal{B}}(x) = \sum_{i\in\mathcal{B}} \Big[\beta_i(x)\, \psi_i(x)\, V_{\mathcal{B}\setminus\{i\}}\big(s(x)\big) \Big] = 0 \quad \text{for all} \quad x \in \mathbb{G}_{\mathcal{B}}\,, \tag{5}$$

where β_i is the degree adjustment polynomial for the ith constraint, in PAIR. Optionally, as was the case with the composition polynomial in section 3.2, the degree of the constraints may be individually adjusted before they are combined, to guarantee that these are treated equally in the RPT. The degree of ψ_i is bounded as $\deg(\psi_i) \leq \deg(C_i) \times \big(|\mathbb{G}| - 1\big)$, hence, for each constraint, the verifier provides a random polynomial [10] β_i over $\mathbb{K}$, that is utilized to adjust the degree of ψ_i. Expressly, the verifier chooses a random polynomial β_i such that

$$\deg(\beta_i) = \Big(\max_{j\in\mathcal{B}} \big\{ \deg(C_j) \big\} - \deg(C_i) \Big) \times \big(|\mathbb{G}| - 1 \big)\,,$$

and, for all $i \in \mathcal{B}$, the new (degree-adjusted) polynomial becomes $\beta_i\psi_i$, with tight public degree-bound

$$\deg(\beta_i\psi_i) \leq \max_{j\in\mathcal{B}} \big\{ \deg(C_j) \big\} \times \big(|\mathbb{G}| - 1 \big)\,.$$

As before, the verifier should guarantee that $\beta_i(\mathbb{G}_i) \cap \{0\} = \emptyset$ for any $i \in \mathcal{B}$. This constitutes an additional interactive step in the overall protocol, and increases the complexity of the arithmetization. Moreover, executing a random combination of the polynomials, decreases the soundness of the arithmetization. As previously explained, randomly combining polynomials under Reed-Solomon membership testing may negatively affect the final proof soundness, as there is a chance of high-degree terms cancelling, in the case of corrupt computational proofs [4].

If the degree adjustment is overlooked, the β_i polynomials will have degree $\deg(\beta_i) = 0$, for all $i \in \mathcal{B}$, making them equivalent to non-zero constants. Note that, recalling the benefits of algebraic linking, completely omitting all β_i (by setting them to the same constant) could be detrimental, as it avoids all randomness in the linear combination of constraint satisfaction problems.

Most importantly (with or without degree adjustments), the function $\psi_{\mathcal{B}}$ displays the following property: $\psi_{\mathcal{B}}(x) = 0 \iff \psi_i(x) = 0$ for $s(x) = i \in \mathcal{B}$. Keeping in mind that the enforcement domain of $\psi_{\mathcal{B}}$ is $\mathbb{G}_{\mathcal{B}}$, the corresponding combined constraint becomes: $\psi_{\mathcal{B}}(x) = 0$ for all $x \in \mathbb{G}_{\mathcal{B}}$.

The combined constraint guarantees the enforcement of the individual AIR constraints in their respective enforcement domains, with the advantage of effectively employing only a single constraint. Nevertheless, $\psi_{\mathcal{B}}$ clearly requires a much more complex formulation, which presents implications for the computational efficiency and protocol security that should not be overlooked. One should also keep in mind that the above merging of constraints is only useful for constraints with disjoint enforcement domains.

4.2 Algebraic placement and routing in PAIR

To obtain a tight degree-bound for this method, note that, if the function $\mathbf{f}$ is indeed a polynomial of degree $\deg(\mathbf{f}) \leq |\mathbb{G}| - 1$, then, in accordance with equation (5), the degree of $\psi_{\mathcal{B}}$ yields

$$\deg(\psi_{\mathcal{B}}) \leq \max_{j \in \mathcal{B}} \big\{\deg(C_j)\big\} \times \big(|\mathbb{G}| - 1\big) + \deg(s) \times \big(|\mathcal{B}| - 1\big)\,.$$

Next, if the function $\psi_{\mathcal{B}}$ has a root for every element of $\mathbb{G}_{\mathcal{B}}$ (an honest prover situation), then

$$x \mapsto Q_{\mathcal{B}}(x) = \frac{\psi_{\mathcal{B}}(x)}{V_{\mathbb{G}_{\mathcal{B}}}(x)} \tag{6}$$

defines a new polynomial suitable for low-degree proximity testing. Hence, the new constraint becomes

$$\deg(Q_{\mathcal{B}}) \leq \max_{j \in \mathcal{B}} \big\{\deg(C_j)\big\} \times \big(|\mathbb{G}| - 1\big) + \deg(s) \times \big(|\mathcal{B}| - 1\big) - |\mathbb{G}_{\mathcal{B}}| = \mathfrak{d}_{Q_{\mathcal{B}}}\,, \tag{7}$$

which utilizes the tightest degree upper bound known by the verifier. Concerning completeness, this constraint holds if the original computational integrity statement is valid. Pertaining to soundness, if random algebraic linking is not utilized, the reverse implication is also true; otherwise, the equivalence is "merely" very likely.

Table 2 summarizes the distribution of knowledge, among the proof intervenients, of the PAIR components. Again, the prover is aware of all information.

Table 2: Knowledge of PAIR components. The comparison applies to all $i \in \mathcal{B}$.

	$\mathbb{F}$	$\mathbb{G}$	$\mathbf{f}$	$\mathbb{G}_i$	$\mathbf{M}_i$	C_i	ψ_i	β_i	s	$Q_{\mathcal{B}}$	$\mathfrak{d}_{Q_{\mathcal{B}}}$
Verifier (public)	yes	yes	**no**	yes	yes	yes	**no**	yes	yes	**no**	yes
Prover (private)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

Comparing with table 1, the components β_i, $Q_{\mathcal{B}}$, and $\mathfrak{d}_{Q_{\mathcal{B}}}$ are analogous to α_i, CP, and $\mathfrak{d}_{\mathrm{CP}}$ respectively, in the AIR method, with the selector column being the only additional element that must be shared. Effectively, this method does not in itself predicate any further compromise concerning the privacy of the execution trace by the prover.

5 Reed-Solomon Proximity Testing: AIR versus PAIR

The degree bounds of the AIR's composition polynomial, CP, and the PAIR's combined constraint polynomial, $Q_{\mathcal{B}}$, will now be compared. Reducing the degree upper bounds is critical in increasing the soundness of the low-degree proximity testing protocols; for instance, the work [14] verifies that increasing the degree bound for low-degree testing increases the FRI soundness error (fixing the batched FRI proximity parameter θ, and rate parameter ρ).

Regarding the FRI in more detail, the soundness of the query phase is determined solely by the proximity parameter and number of query phase iterations (one for each random sample at layer zero). On the other hand, the FRI commit phase error accumulates, over all FRI layers, the systematic error of the degree respecting projection rounds. In other words, if the fractional Hamming distance between the original function (at layer zero) and the relevant low-degree codes surpasses the proximity parameter, then the distance from the oracles produced during the commit phase (the subsequent FRI layers) to the corresponding low-degree codes also exceeds the proximity parameter, unless with probability bounded above by the commit phase error. The commit phase error depends directly on the degree bound for RPT [Theorem 8.2 of 6].

Therefore, comparing the degree bounds of CP and $Q_{\mathcal{B}}$ is essential to determine the most adequate option for optimized Reed-Solomon proximity testing in STARKs. To have a fair comparison, the C_i constraints used in both cases will be the same and indexed by the set $\mathcal{B}$.

5.1 Deriving $\mathfrak{d}_{Q_{\mathcal{B}}} \geq \mathfrak{d}_{\mathrm{CP}}$

To state the relation between the degree bounds, lemma 5.1 is required. Also, recall that $s(\mathbb{G}_{\mathcal{B}}) = \mathcal{B}$ and $\mathbb{G}_i = \{x \in \mathbb{G} : s(x) = i\}$.

Lemma 5.1. *Let s be a polynomial over a finite field $\mathbb{F}$ of degree $\deg(s) > 0$, and $\mathbb{G}_i$ a subset of $\mathbb{F}$ defined as $\mathbb{G}_i = \{x : s(x) = i\}$. Then $|\mathbb{G}_i| \leq \deg(s)$ for all $i \in \mathbb{F}$.*

Proof. By contradiction, if this statement were false, then:

$$\exists\, i \in \mathbb{F} :\ \big|\{x : s(x) = i\}\big| > \deg(s) \quad \iff \quad \big|\{x : s(x) - i = 0\}\big| > \deg(s). \tag{8}$$

Because the polynomial $x \mapsto s(x) - i$ has degree $\deg(s)$, it cannot have more than $\deg(s)$ roots[7], rendering inequality (8) impossible. Hence, lemma 5.1 is true. □

Theorem 5.2. *The degree bound of the PAIR combined constraint polynomial is larger than or equal to the degree bound of the AIR composition polynomial. Expressly:* $\mathfrak{d}_{Q_\mathcal{B}} \geq \mathfrak{d}_{\mathrm{CP}}$.

Proof. Firstly, the difference between the degree bounds may be expanded as

$$\begin{aligned}
\mathfrak{d}_{Q_\mathcal{B}} - \mathfrak{d}_{\mathrm{CP}} &= \Big[\max_{j\in\mathcal{B}}\big\{\deg(C_j)\big\} \times \big(|\mathbb{G}| - 1\big) + \deg(s) \times \big(|\mathcal{B}| - 1\big) - |\mathbb{G}_\mathcal{B}|\Big] - \max_{j\in\mathcal{B}}\big\{\deg(C_j) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}_j|\big\} = \\
&= \deg(C_i) \times \big(|\mathbb{G}| - 1\big) + \deg(s) \times \big(|\mathcal{B}| - 1\big) - |\mathbb{G}_\mathcal{B}| - \Big[\deg(C_i) \times \big(|\mathbb{G}| - 1\big) - |\mathbb{G}_i|\Big] = \\
&= \deg(s) \times \big(|\mathcal{B}| - 1\big) + |\mathbb{G}_i| - |\mathbb{G}_\mathcal{B}| \quad \text{where } i \text{ satisfies} \quad \mathfrak{d}_{Q_i} = \mathfrak{d}_{\mathrm{CP}} \quad \text{and} \quad \deg(C_i) = \max_{j\in\mathcal{B}}\big\{\deg(C_j)\big\}.
\end{aligned}$$

Note that theorem 3.1 guarantees the existence of such $i \in \mathcal{B}$. Moreover, since $\mathbb{G}_i \cap \mathbb{G}_j = \emptyset$ for all $i \neq j$, the enforcement domain of the combined constraint is the disjoint union of the AIR enforcement domains, and consequently

$$\mathbb{G}_\mathcal{B} = \bigsqcup_{j\in\mathcal{B}} \mathbb{G}_j \implies |\mathbb{G}_\mathcal{B}| = \sum_{j\in\mathcal{B}} |\mathbb{G}_j|\,.$$

Hence, by applying lemma 5.1, for $i \in \mathcal{B}$ and $\deg(s) > 0$ one has

$$\begin{aligned}
|\mathbb{G}_\mathcal{B}| = \sum_{j\in\mathcal{B}} |\mathbb{G}_j| = \sum_{j\in\mathcal{B}\setminus\{i\}} |\mathbb{G}_j| + |\mathbb{G}_i| \leq \sum_{j\in\mathcal{B}\setminus\{i\}} \deg(s) + |\mathbb{G}_i| = \deg(s)\big(|\mathcal{B}| - 1\big) + |\mathbb{G}_i| \quad \iff \\
\iff \quad \deg(s)\big(|\mathcal{B}| - 1\big) + |\mathbb{G}_i| - |\mathbb{G}_\mathcal{B}| = \mathfrak{d}_{Q_\mathcal{B}} - \mathfrak{d}_{\mathrm{CP}} \geq 0 \iff \mathfrak{d}_{Q_\mathcal{B}} \geq \mathfrak{d}_{\mathrm{CP}}\,.
\end{aligned}$$

Finally, if $\deg(s) = 0$ then the polynomial is constant, hence $s(\mathbb{G}_\mathcal{B}) = \{i\} = \mathcal{B}$ for some i. In that case, $\mathbb{G}_i = \{x \in \mathbb{G} : s(x) = i\} = \mathbb{G}_\mathcal{B}$ and $|\mathcal{B}| = 1$. Therefore,

$$\deg(s)\big(|\mathcal{B}| - 1\big) + |\mathbb{G}_i| - |\mathbb{G}_\mathcal{B}| = 0 \geq 0 \implies \mathfrak{d}_{Q_\mathcal{B}} \geq \mathfrak{d}_{\mathrm{CP}}\,.$$

□

Theorem 5.2 thus shows that the degree bound for the PAIR low-degree constraint, using $Q_\mathcal{B}$, will always be greater than or equal to the degree bound for the AIR low-degree constraint, using CP, regardless of the total amount of disjoint constraints a particular execution trace is subject to. This includes the example of two elementary constraints for addition and multiplication [12, 18]. Therefore, in terms of optimizing low-degree proximity testing, it is not advantageous to construct a selector function to merge all constraints into a single one, rather than using separate AIR constraints with corresponding enforcement domains.

Concerning security implications for Reed-Solomon proximity testing, fixing all other parameters, AIR thus results in a better or equal soundness than PAIR.

5.2 Slicing for Batched RPT with Lower Rate

It is possible to lower the degree of the codes for Reed-Solomon proximity testing in both AIR and PAIR by using the slicing technique (also known as interlacing or segmenting). Namely, for a polynomial p of degree $\deg(p) \leq \mathfrak{d}_p$, and slicing step c, it is possible to construct new polynomials p_i with $i \in \{0, \ldots, c-1\}$ of degree $\deg(p_i) \leq \mathfrak{d}_p/c$, such that

$$p(x) = \sum_{i=0}^{c-1} x^i\, p_i(x^c)\,.$$

[7] This property is proved in corollary 3 of theorem 16.2 of the document [13].

Subsequently, using random elements $z_i \in \mathbb{K}$ provided by the verifier, a new polynomial $\widetilde{p}$ is built as a random linear combination of the p_i polynomials, yielding

$$x \mapsto \widetilde{p}(x) = \sum_{i=0}^{c-1} z_i \, p_i(x) \quad \text{with degree} \quad \deg(\widetilde{p}) \leq \mathfrak{d}_p / c \, .$$

Finally, by converting high degree-bound polynomials into polynomials of lower degree through slicing it is possible to perform batched RPT [14, 6] with a lower rate. Noticeably, this approach is common in the industry and is performed, for instance, in [21]. Using public-coins from the verifier to randomly combine the p_i polynomials yields a new Reed-Solomon proximity test with a smaller degree-bound, $\mathfrak{d}_p/c$.

Naturally, the analysis of section 5.1 does not consider the application of this procedure since it may be performed for either AIR or PAIR. Nonetheless, slicing might require significant computation and introduces another interaction between the prover and the verifier, which, again, can be simulated by a non-interactive procedure [7].

6 Tweaking the PAIR Selector—Optimization of the Bound

By analyzing equation (7), one observes that $\mathfrak{d}_{Q_\mathcal{B}}$ varies linearly with the cardinality of the image of $\mathbb{G}_\mathcal{B}$ through the selector s; expressly, it increases linearly with $|\mathcal{B}|$. This section will provide a method for mitigating this factor in the PAIR selector, without compromising the previously defined AIR.

Instead of using a single selector function, one could append multiple selector columns to the trace—still derived from (4)—so that these would be combined to uniquely represent a selecting value as before. The advantage here would arise from the decrease in the size of the images of each selector, albeit the added constraint columns would reflect in further processing and memory requirements.

6.1 Example: Binary Selection of 2^3 Constraints

Suppose the case of merely eight disjoint ψ_i constraints, that is, $|\mathcal{B}| = 8$. To subserve representation, admit that $\mathcal{B} = \{0, \ldots, 7\}$—any different set of 8 constraint could simply be reindexed to this one. Without loss of generality, PAIR may be structured as in table 3, where k indexes the elements of the trace evaluation domain (not necessarily in a consecutive manner). The selector column s shows the approach of the regular PAIR described so far, expressing the disjoint application of different constraints; the remaining selectors s_0, s_1 and s_2 uniquely encode s in a binary representation, forming a separate but equivalent preprocessed trace.

Table 3: Preprocessed execution trace tables subject to $|\mathcal{B}| = 8$ different constraints, with the regular selector, and the binarily decomposed selector. The rows are ordered as $0 \leq k_0 < k_1 < \ldots < k_7 < |\mathbb{G}|$, without loss of generality, and $x = g^k$.

		regular		base $a = 2$			
Constraint	k	$\mathbf{f}(x)$	s	$\mathbf{f}(x)$	s_2	s_1	s_0
$\psi_0(x) = 0$	k_0	$\cdots$	0	$\cdots$	0	0	0
$\psi_1(x) = 0$	k_1	$\cdots$	1	$\cdots$	0	0	1
$\psi_2(x) = 0$	k_2	$\cdots$	2	$\cdots$	0	1	0
$\psi_3(x) = 0$	k_3	$\cdots$	3	$\cdots$	0	1	1
$\psi_4(x) = 0$	k_4	$\cdots$	4	$\cdots$	1	0	0
$\psi_5(x) = 0$	k_5	$\cdots$	5	$\cdots$	1	0	1
$\psi_6(x) = 0$	k_6	$\cdots$	6	$\cdots$	1	1	0
$\psi_7(x) = 0$	k_7	$\cdots$	7	$\cdots$	1	1	1

Regarding the regular PAIR method: (i) the low-degree extension applies only to one selector function, and (ii) the polynomial of the APR combined constraint, as shown in equation (7), has a degree bound $\mathfrak{d}_{Q_\mathcal{B}} = \max_{j \in \mathcal{B}} \left\{ \deg(C_j) \right\} \times |\mathbb{G}| + 7 \times \deg(s) - |\mathbb{G}_\mathcal{B}|$. On the other hand, when decomposing a single selector into a multitude of binary selectors, the low-degree extension occurs for more functions (three instead of one, in the example of table 3), and the combined constraint polynomial may be expressed as:

$$\begin{aligned} \psi'_\mathcal{B} = (1 - s_2) \cdot \Big[(1 - s_1) \cdot \big\{ (1 - s_0) \cdot \psi_0 + s_0 \cdot \psi_1 \big\} + s_1 \cdot \big\{ (1 - s_0) \cdot \psi_2 + s_0 \cdot \psi_3 \big\} \Big] \\ + s_2 \ \cdot \Big[(1 - s_1) \cdot \big\{ (1 - s_0) \cdot \psi_4 + s_0 \cdot \psi_5 \big\} + s_1 \cdot \big\{ (1 - s_0) \cdot \psi_6 + s_0 \cdot \psi_7 \big\} \Big] . \end{aligned}$$

Recalling equation (6), the difference between the degree bounds of $Q_{\mathcal{B}}$ and $Q'_{\mathcal{B}} = \psi'_{\mathcal{B}}/V_{\mathbb{G}_{\mathcal{B}}}$ will be

$$\mathfrak{d}_{Q_{\mathcal{B}}} - \mathfrak{d}_{Q'_{\mathcal{B}}} = 7 \times \deg(s) - 3 \times \overline{\deg(s_j)}\,,$$

where the overline denotes the average. Therefore, for this particular instance, there is an improvement in the degree-bound if the average degree of the three binary selectors is smaller than $7/3 \approx 2.3$ of the degree of the regular selector. In fact, one would expect the degree of s_0, s_1 and s_3 to be larger than the degree of s, due to the greater amount of repeated values, which, by lemma 5.1, increases the lower bound for the degree of the interpolating polynomial. So when would this decomposition method be serviceable?

6.2 Generalizing the Representation Base and Number of Constraints

When utilizing a generic representation base a, each of the multiple selector functions maps the union of the constraint enforcement domains, $\mathbb{G}_{\mathcal{B}}$, to the set $\mathcal{A} = \{0, \ldots, a-1\}$. To represent $|\mathcal{B}|$ disjoint constraints, the number of selector columns, n_s, will be given by $n_s = \lceil \log_a |\mathcal{B}| \rceil$. Hence, one may express the corresponding PAIR combined constraint as

$$\psi'_{\mathcal{B}}(x) = \sum_{i \in \mathcal{B}} \left[\beta_i(x)\ \psi_i(x) \prod_{j=0}^{n_s-1} V_{\mathcal{A}\setminus\{i_j\}}\big(s_j(x)\big) \right] = 0 \quad \text{for all} \quad x \in \mathbb{G}_{\mathcal{B}} \quad \text{with} \quad s_j(x) = \sum_{i \in \mathcal{B}} i_j \sum_{z \in \mathbb{G}_i} l(x|z)\,,$$

where $i_j \in \mathcal{A}$ is the jth digit (starting from zero) in the representation of i base a [8]. This leads to the degree bound

$$\deg(\psi'_{\mathcal{B}}) \leq \max_{j \in \mathcal{B}} \big\{\deg(C_j)\big\} \times \big(|\mathbb{G}| - 1\big) + \big(|\mathcal{A}| - 1\big) \times \sum_{j=0}^{n_s-1} \deg(s_j)\,.$$

Therefore, considering $Q'_{\mathcal{B}} = \psi'_{\mathcal{B}}/V_{\mathbb{G}_{\mathcal{B}}}$, for the APR constraint one has

$$\deg(Q'_{\mathcal{B}}) \leq \max_{j \in \mathcal{B}} \big\{\deg(C_j)\big\} \times \big(|\mathbb{G}| - 1\big) + \big(a - 1\big) \times \lceil \log_a |\mathcal{B}| \rceil \times \overline{\deg(s_j)} - |\mathbb{G}_{\mathcal{B}}| = \mathfrak{d}_{Q'_{\mathcal{B}}}\,. \tag{9}$$

Noticeably, it is not trivial to guarantee a degree-bound improvement since the value $\overline{\deg(s_j)}$ has a higher lower bound than $\deg(s)$, induced by the pigeonhole principle. Namely, it implies that

$$\big|\{x \in \mathbb{G}_{\mathcal{B}} : s_j(x) = i\}\big| \geq \left\lceil \frac{|\mathbb{G}_{\mathcal{B}}|}{|s_j(\mathbb{G}_{\mathcal{B}})|} \right\rceil \quad \text{for some} \quad i \in s_j(\mathbb{G}_{\mathcal{B}}) \quad \text{for all} \quad j \in \{0, \ldots, n_s - 1\}\,.$$

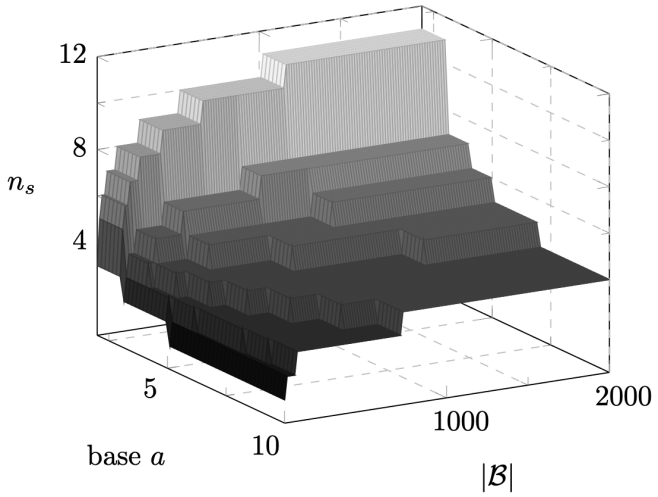
Therefore, by lemma 5.1, one has $\deg(s_j) \geq \lceil |\mathbb{G}_{\mathcal{B}}|/a \rceil$, since $s_j(\mathbb{G}_{\mathcal{B}}) \subset \mathcal{A}$. Not much else can be predicted about the degree of general selector functions. However, comparing equations (7) and (9), establishes that this last approach will lead to a lower degree bound for Reed-Solomon proximity testing whenever the average degree of the selectors s_j is below a threshold ζ that depends on the base a and number of constraints $|\mathcal{B}|$. Expressly,

$$\mathfrak{d}_{Q'_{\mathcal{B}}} < \mathfrak{d}_{Q_{\mathcal{B}}} \iff \frac{\overline{\deg(s_j)}}{\deg(s)} < \frac{\ln(a)}{a-1} \frac{|\mathcal{B}| - 1}{\ln\big(|\mathcal{B}|\big)} = \zeta\,.$$

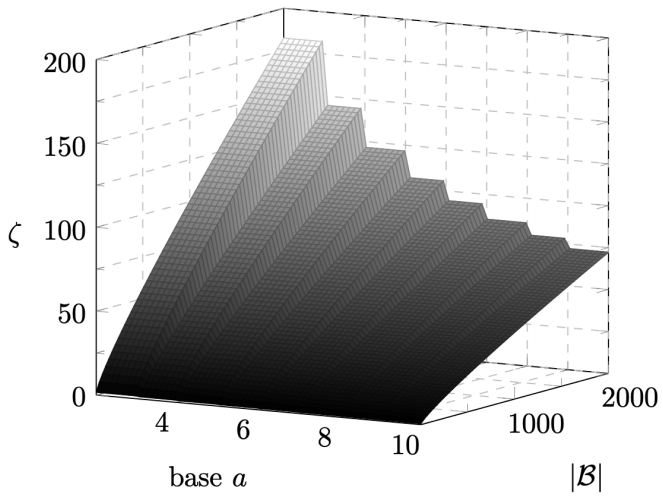
Noting now that the function $x \mapsto \ln(x)/(x-1)$ is strictly decreasing, one infers that, if the base a is small, there is a larger margin for improvement of the low-degree bound. On the other hand, increasing $|\mathcal{B}|$ has the opposite effect on ζ, which makes this approach more desirable when employing a larger amount of constraints.

Figure 1a illustrates that decreasing the representation base a increases the number of selector functions that must undergo the low-degree extension, an undesirable effect. Conversely, it also increases the low-degree improvement threshold ζ, an advantageous impact (figure 1b). To optimize computational performance, one must balance these two opposite trends: whether to incur in further arithmetization expenses by adding selector columns (decrease a) while optimizing the degree bound for the subsequent Reed-Solomon proximity testing, or do the reciprocal. The number of constraints, $|\mathcal{B}|$, must also be taken into consideration in this optimization, inducing effects opposite to those of a. Nonetheless, besides affecting the PAIR, changing $|\mathcal{B}|$ has other significant effects on arithmetization, namely on the degree of the AIR polynomials, C_i, and on the size of the execution trace.

[8] As an example, in base $a = 3$, for the $i = 7$ element of the summation, and the $j = 1$ product element, the vanishing polynomial is $x \mapsto V_{\mathcal{A}\setminus\{2\}}(x) = x(x-1)$, since $7 = 2 \times 3^1 + 1 \times 3^0 = (21)_3$.



(a) Number of columns, $n_s = \lceil \log_a |\mathcal{B}| \rceil$.



(b) Threshold for improvement of $\mathfrak{d}_{Q'_{\mathcal{B}}}$ for RPT.

Figure 1: Influence of the base a and number of constraints $|\mathcal{B}|$ on the number of selector columns n_s and on ζ, the threshold for improvement of the degree bound for Reed-Solomon proximity testing, $\mathfrak{d}_{Q'_{\mathcal{B}}}$.

Appendix A compares the specific case of 9 disjoint constraints, when represented by either base 2 or base 3 selectors, analysing the resulting degree bounds.

7 Scalability and Complexity

Ensuring that STARKs enhance scalability by efficiently verifying computations on large batches requires taking into account the computational requirements of their underlying components. This section briefly analyzes the computational demands of the two protocols—PAIR and AIR—and explores how they impact performance optimization, specifically, the low-degree extension (LDE).

The LDE of $\mathbf{f}$, CP and $Q_{\mathcal{B}}$ is the prover-executed evaluation of said functions over the evaluation domain $w\mathbb{H}$, with $\mathbb{G} < \mathbb{H}$ and $w \in \mathbb{F}^{\times}$—it is common to enforce a proper coset $w\mathbb{H}$ with $w \notin \mathbb{H}$. The evaluation domain for the LDE has these properties for multiple reasons:

(i) $\mathbb{G} < \mathbb{H}$, the proper subgroup specification

  (a) facilitates the LDE by allowing the use of efficient FFT-based algorithms for multi-point polynomial evaluation (section 3.4 of [21]), and

  (b) introduces a blow-up factor (a rate reduction) that increases soundness in Reed-Solomon proximity testing (figure 1 of [8]);

(ii) $\mathbb{G} \cap w\mathbb{H} = \emptyset$, the disjunction condition

  (a) prevents the verifier from querying the rational functions from the APR at undefined points, namely, those in the enforcement domains $\mathbb{G}_i$ (section 3.3.3 of [21]), and

  (b) ensures that the verifier will not sample the private execution trace in query phases, promoting zero-knowledge (section 3.7 of [14]).

The LDE is a relevant performance assessment point because it is the dominant function evaluation step in the arithmetization phase, prior to RPT, and occurs in the entirety of the evaluation domain, $w\mathbb{H}$, yielding a significant computational effort [5]. Preceding the LDE, the arithmetization involves algebraic transformations that can largely exist in abstraction with minimal computational demands. As a remark, given a predetermined AIR instance, the arithmetization phase places the majority of the computational workload on the prover, as the verifier's role is limited to supplying randomness, at most (in the case of an IOP).

There are several methods available for performing the low-degree extension of a polynomial p over $\mathbb{F}$. One option is executing multiple fast Fourier transforms [22], each applied to an array of length $|\mathbb{A}|$, where $\mathbb{A}$ is the smallest multiplicative subgroup of $\mathbb{H}$ with order greater than $\deg(p)$. In total, considering all the cosets of $\mathbb{A}$ in

$\mathbb{H}$, one must perform $|\mathbb{H}|/|\mathbb{A}|$ FFTs of size $|\mathbb{A}|$. When evaluating cosets instead of groups, the array of polynomial coefficients of p needs to be modified with proper adjustment factors. Namely, (i) regarding the cosets of $\mathbb{A}$ in $\mathbb{H}$, the factors will vary between all FFTs and between all coefficients, and (ii) if $w\mathbb{H}$ is a proper coset of $\mathbb{H}$, the FFT requires additional factors that differ from coefficient to coefficient. If these factors are precomputed, the adjustment operations total $O\big(\mathbb{H}\big)$ field multiplications. Futhermore, each fast Fourier transform outputs $|\mathbb{A}|$ new point evaluations and has complexity $O\big(|\mathbb{A}|\log(|\mathbb{A}|)\big)$, resulting in a total complexity of $O\big(|\mathbb{H}|\log(|\mathbb{A}|)\big)$ for the LDE, after acknowledging all cosets of $\mathbb{A}$ in $\mathbb{H}$. Employing the FFT is a nicely scalable solution for computing the low-degree extension of p.

If the array of polynomial coefficients of p is sparse, then polynomial evaluation is much simpler, and, at best, computing a single point evaluation scales as $O\big(\log(\deg(p))\big)$. For multi-point evaluation, the sparsity of p will also accelerate the FFT. Another fringe optimal case occurs when the function is periodic over $w\mathbb{H}$, expressly if there exists $y \in \mathbb{H} \setminus \{1\}$ such that $p(yx) = p(x)$ for all $x \in w\mathbb{H}$. At best, the order[9] of y will be $|\mathbb{H}|/\deg(p)$, resulting in a period of the same length, and hence, only that many points need to be function-evaluated. These observations might yield some optimizations for polynomial evaluation, when applicable.

After the arithmetization, one will undertake the Reed-Solomon proximity testing. However, the PAIR method may experience increased complexity or reduced soundness compared to AIR, due to its generally higher low-degree bounds. Specifically, for FRI using the PAIR method, the higher low-degree bounds may result in a larger soundness error and more layers per batch, which adds to the complexity. Consequently, to meet the fixed security requirements, the Reed-Solomon proximity testing proof-size will increase for the PAIR method compared to AIR.

7.1 Direct Comparison of Components

To optimize the low-degree extension of the polynomials CP and $Q_{\mathcal{B}}$, it is crucial to have a clear understanding of the individual components of each function, so that one may develop efficient procedures for this function evaluation step. It is essential to avoid omitting details in the mathematical expressions, as these could be significant in the optimization process. To reiterate, the formulation for the composition polynomial is

$$x \mapsto \mathrm{CP}(x) = \sum_{i \in \mathcal{B}} \left[\frac{\alpha_i(x)\, \psi_i(x)}{V_{\mathbb{G}_i}(x)} \right],$$

and for the PAIR, the polynomial of the APR constraint is

$$x \mapsto Q_{\mathcal{B}}(x) = \frac{1}{V_{\mathbb{G}_{\mathcal{B}}}(x)} \times \sum_{i \in \mathcal{B}} \left[\beta_i(x)\ \psi_i(x) \prod_{j=0}^{n_s-1} V_{\mathcal{A}\setminus\{i_j\}}\big(s_j(x)\big) \right].$$

Table 4 illustrates which components must be computed in the AIR and PAIR protocols. Again, for the comparison to be meaningful, the AIR and PAIR refer to the same set of disjoint constraints.

Table 4: Comparison of elements that must be computed between the AIR and PAIR methods. The regular PAIR is derived by defining $\mathcal{A} = \mathcal{B} = \{0, \ldots, |\mathcal{B}| - 1\}$, which implies $n_s = 1$.

Compute	**AIR**	**PAIR**
$\alpha_i(x)$ for all $x \in w\mathbb{H}$ for all $i \in \mathcal{B}$	optional	no
$V_{\mathbb{G}_i}(x)$ for all $x \in w\mathbb{H}$ for all $i \in \mathcal{B}$	yes	no
$\psi_i(x)$ for all $x \in w\mathbb{H}$ for all $i \in \mathcal{B}$	yes	yes
$\beta_i(x)$ for all $x \in w\mathbb{H}$ for all $i \in \mathcal{B}$	no	optional
$V_{\mathcal{A}\setminus\{i_j\}}(y)$ for all $y \in s_j(w\mathbb{H})$ for all $j \in [0, n_s)$ for all $i \in \mathcal{B}$	no	yes
$s_j(x)$ for all $x \in w\mathbb{H}$ for all $j \in [0, n_s)$	no	yes
$V_{\mathbb{G}_{\mathcal{B}}}(x)$ for all $x \in w\mathbb{H}$	no	yes

The utilization of degree adjustment polynomials, α_i and β_i, is not a requirement in arithmetization. However, their usage can be customized to fit specific needs, allowing for flexibility in degree adjustment. Depending on the

[9] The multiplicative order of y is the smallest natural number k such that $y^k = 1$.

arithmetization instance, all α_i can either have a higher degree than all β_i, the reverse, or the degrees can vary in between. The construction of the degree adjustment polynomials will depend on the degrees of the C_i polynomials and the sizes of the enforcement domains.

The main computational disadvantage of the LDE in PAIR arises from the necessity of interpolating and evaluating the selector polynomials. Also, if the method described in section 6 is used, then one may need to consider the system requirements for supporting the extended trace table formed by appending multiple selector columns. Finally, PAIR requires the evaluation of the vanishing polynomials, $V_{\mathcal{A}\setminus\{i_j\}}$, for all the constraints (indexed by i) and all the selectors columns (indexed by j). Naturally, AIR without selectors does not have these issues.

Conversely, the main drawback of the AIR method will be its necessity to compute the vanishing polynomial, $\mathrm{V}_{\mathbb{G}_i}$, for each individual constraint. In contrast, the PAIR method combines all these into a single "larger" constraint, requiring the computation of only one vanishing polynomial. As a remark, evaluating a vanishing polynomial becomes much easier if its set of roots is very close to the set of a multiplicative group [20]—for example, if $\mathbb{G}_{\mathcal{B}} = \mathbb{G} \setminus \{g^2\}$, then $V_{\mathbb{G}_{\mathcal{B}}}(x) = (x^{|\mathbb{G}|} - 1)/(x - g^2)$, which avoids explicitly computing a lengthy product.

It is important to examine the specific application context of each project when evaluating the advantages and disadvantages mentioned above. Determining how these subtle computational differences impacts the actual performance of an implementation requires (i) criteria to balance tasks between the prover and the verifier, (ii) an evaluation of said chores regarding processing power, memory and network requirements, with suitable benchmark testing, and (iii) real-world practical testing. Once the computational requirements for the arithmetization stage are taken into account, it is also relevant to conduct a similar analysis for the RPT. A methodical approach must be adopted to study how the varying degree bounds of AIR and PAIR impact the complexity and proof-length of the RPT, while maintaining a fixed level of soundness. The final decisions balancing the two approaches should be made based on the results of system integration testing. Further exploration of this topic would be valuable for advancing research in this area.

8 Transitioning from PAIR to AIR

This section will provide some general guidelines on how one could easily convert a PAIR implementation into a corresponding AIR, with each constraint individually defined alongside its enforcement domain, rather then the unique combined PAIR constraint polynomial, $Q_{\mathcal{B}}$.

Table 4 shows that there are at most two components of the AIR that are not components of the PAIR, expressly: (i) the vanishing polynomials of the enforcement domains, and (ii) the optional degree-adjustment polynomials, which require public randomness. As in section 4, when the PAIR approach is initially undertaken, the selector values must be publicly defined along the execution trace, since these determine the constraint being enforced at the corresponding execution trace row. To initiate the transitioning process, these selector values are used to generate the enforcement domains, $\mathbb{G}_i$, as level-sets of s. Once this is done, the procedure is akin to section 3, as follows.

(i) Perform the APR on the individual constraint polynomials with the newly defined $\mathbb{G}_i$'s.

(ii) Conduct the ALI reduction with public-randomness provided by the verifier [10] to build the composition polynomial.

Overall, the transition from PAIR to AIR is simpler than the reverse, with the only major addition of an optional interaction step.

9 Hybrid Approach

In section 5 it was shown that the degree bound for the RPT after an AIR will always be less than or equal to the degree bound of the corresponding (regular) PAIR. However, it might be possible to enjoy some of the advantages of PAIR (like those suggested in section 7) without deteriorating the degree bound. Specifically, a hybrid strategy would avoid this issue, if there exists a proper subset of constraints which combined using the PAIR approach leads to a degree bound which is at most the degree bound of the composition polynomial CP. In that case, the PAIR procedure could be used without affecting the final low-degree bound.

[10] This interaction can be simulated by a non-interactive procedure [7].

Formally, if there exists $\mathcal{E} \subset \mathcal{B}$ such that $Q_{\mathcal{E}}$, the PAIR combined constraint of the AIR constraints index by $\mathcal{E}$, has degree

$$\deg(Q_{\mathcal{E}}) \leq \mathfrak{d}_{Q_{\mathcal{E}}} \leq \max_{j \in \mathcal{B}} \{\mathfrak{d}_{Q_j}\} = \mathfrak{d}_{\mathrm{CP}}\,, \quad \text{then} \quad x \mapsto \widetilde{\mathrm{CP}}(x) = \alpha_{\mathcal{E}}(x)\, Q_{\mathcal{E}}(x) + \sum_{i \in \mathcal{B} \setminus \mathcal{E}} \alpha_i(x)\, Q_i(x)$$

defines a new composition polynomial with the same degree bound, namely $\max_{j \in \mathcal{B}} \{\mathfrak{d}_{Q_j}\} = \mathfrak{d}_{\mathrm{CP}} = \mathfrak{d}_{\widetilde{\mathrm{CP}}}$. Additionally, $\alpha_{\mathcal{E}}$ is a random polynomial chosen by the verifier with degree $\deg(\alpha_{\mathcal{E}}) = \mathfrak{d}_{\mathrm{CP}} - \mathfrak{d}_{Q_{\mathcal{E}}} \geq 0$, if performing degree-adjustment. Contrarily, the degree of $\alpha_{\mathcal{E}}$ is set to zero.

To sum things up, one could build a composition polynomial using the PAIR combined constraint for better performance—mostly, by reducing the amount of vanishing polynomials in the final RPT assignment—while still optimizing low-degree proximity testing by benefiting of the same low-degree bound as in the corresponding AIR.

10 Conclusions

This paper discussed two constraint arithmetization methods: AIR and its preprocessed variant PAIR, that employs selector columns for constraint enforcement. The document explained the application of these methods within a proving system and derived the degree bounds for Reed-Solomon proximity testing (RPT), for either approach.

The study showed that using PAIR increases the degree bound for RPT—due to the need to interpolate and represent the selectors as polynomials—which can lessen the soundness or complexity of the low-degree testing. However, the paper also explored the possibility of reducing the degree bound of PAIR by representing the selector using multiple selector columns, albeit with additional computational demands in the arithmetization phase.

While PAIR simplifies constraint enforcement, it can be easily translated to AIR, and both methods can be used simultaneously for disjoint sets of constraints. This study suggests that PAIR may have computational advantages over AIR, but system testing with benchmarks is necessary to determine the application-specific superiority of each method.

References

[1] L. Babai and L. Fortnow. "Arithmetization: A New Method in Structural Complexity Theory." In: *Computational Complexity* 1 (Mar. 1991), pages 41–66. DOI: 10.1007/BF01200057.

[2] E. Ben-Sasson, I. Ben-Tov, A. Chiesa, A. Gabizon, D. Genkin, M. Hamilis, E. Pergament, M. Riabzev, M. Silberstein, E. Tromer, and M. Virza. *Computational integrity with a public random string from quasi-linear PCPs*. IACR Cryptology ePrint Archive, Paper 2016/646. 2016. URL: https://eprint.iacr.org/2016/646.

[3] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Edited by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Volume 107. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9018.

[4] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. IACR Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046.

[5] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. "Scalable zero knowledge with no trusted setup". In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, Proceedings, Part III 39*. Springer. Aug. 2019, pages 701–732.

[6] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf. *Proximity gaps for Reed-Solomon codes*. Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 83 (2020). July 2021. URL: https://eprint.iacr.org/2020/654.

[7] E. Ben-Sasson, A. Chiesa, and N. Spooner. "Interactive oracle proofs". In: *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*. Springer. 2016, pages 31–60.

[8] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. *DEEP-FRI: Sampling Outside the Box Improves Soundness*. IACR Cryptology ePrint Archive, Paper 2019/336. 2019. URL: https://eprint.iacr.org/2019/336 (visited on 03/30/2023).

[9] E. Ben-Sasson and M. Sudan. "Short PCPs with polylog query complexity". In: *SIAM Journal on Computing* 38.2 (2008), pages 551–607.

[10] A. T. Bharucha-Reid and M. Sambandham. *Random polynomials: Probability and mathematical statistics: a series of monographs and textbooks*. Academic Press, 2014.

[11] A. Fiat and A. Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Proceedings on Advances in Cryptology—CRYPTO '86*. Santa Barbara, California, USA: Springer-Verlag, 1987, pages 186–194.

[12] A. Gabizon. *From AIRs to RAPs - how PLONK-style arithmetization works*. 2021. URL: https://hackmd.io/@aztec-network/plonk-arithmetiization-air (visited on 03/30/2023).

[13] J. A. Gallian. *Contemporary abstract algebra*. 7th edition. Brooks-Cole/Cengage Learning, Belmont, CA, 2010.

[14] U. Haböck. *A summary on the FRI low degree test*. IACR Cryptology ePrint Archive, Paper 2022/1216. 2022. URL: https://eprint.iacr.org/2022/1216 (visited on 03/30/2023).

[15] A. K. Kaw, E. K. Kalu, and D. Nguyen. *Numerical Methods With Applications*. University of South Florida, July 2011.

[16] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the ACM (JACM)* 39.4 (Oct. 1992), pages 859–868.

[17] H. Masip-Ardevol, M. Guzmán-Albiol, J. Baylina-Melé, and J. L. Muñoz-Tapia. *eSTARK: Extending STARKs with Arguments*. Cryptology ePrint Archive, Paper 2023/474. Apr. 2023. URL: https://eprint.iacr.org/2023/474.

[18] Not a Monad Tutorial. *Arithmetization schemes for ZK-SNARKs*. Jan. 2023. URL: https://www.notamonadtutorial.com/arithmetization-schemes-for-zk-snarks/ (visited on 03/30/2023).

[19] Polygon Miden. *AirScript Documentation*. Feb. 2023. URL: https://0xpolygonmiden.github.io/air-script/ (visited on 03/30/2023).

[20] StarkWare Team. *Arithmetization II - "We Need To Go Deeper"*. Mar. 2019. URL: https://medium.com/starkware/arithmetization-ii-403c3b3f4355 (visited on 03/30/2023).

[21] StarkWare Team. *ethSTARK Documentation – Version 1.1*. IACR Cryptology ePrint Archive, Paper 2021/582. Jan. 2021. URL: https://eprint.iacr.org/2021/582.

[22] A. Terras. *Fourier Analysis on Finite Groups and Applications*. London Mathematical Society Student Texts. Cambridge University Press, 1999. DOI: 10.1017/CBO9780511626265.

Appendix A

Example: Choosing the PAIR Selector Base for 3^2 Constraints

This appendix expands on the case of 9 disjoint constraints, to illustrate how one could still choose different bases for the selector columns, while balancing the setbacks and benefits of each.

Table 5 shows this decomposition in both base 2 and 3. Because $9 = 3^2$, the base 3 selector needs only 2 columns, while the base 2 selector requires an entire new column to be able to encode just one additional constraint compared to table 3, adding up to 4 total columns that must be appended to the trace.

Table 5: Preprocessed execution trace tables subject to 9 different constraints, with the regular selector, when the selector base is $a = 3$, and when $a = 2$. The rows are ordered as $0 \leq k_0 < k_1 < \ldots < k_8 < |\mathbb{G}|$, without loss of generality, and $x = g^k$.

		regular		base $a = 3$			base $a = 2$				
Constraint	k	$\mathbf{f}(x)$	s	$\mathbf{f}(x)$	s_1	s_0	$\mathbf{f}(x)$	s_3	s_2	s_1	s_0
$\psi_0(x) = 0$	k_0	$\cdots$	0	$\cdots$	0	0	$\cdots$	0	0	0	0
$\psi_1(x) = 0$	k_1	$\cdots$	1	$\cdots$	0	1	$\cdots$	0	0	0	1
$\psi_2(x) = 0$	k_2	$\cdots$	2	$\cdots$	0	2	$\cdots$	0	0	1	0
$\psi_3(x) = 0$	k_3	$\cdots$	3	$\cdots$	1	0	$\cdots$	0	0	1	1
$\psi_4(x) = 0$	k_4	$\cdots$	4	$\cdots$	1	1	$\cdots$	0	1	0	0
$\psi_5(x) = 0$	k_5	$\cdots$	5	$\cdots$	1	2	$\cdots$	0	1	0	1
$\psi_6(x) = 0$	k_6	$\cdots$	6	$\cdots$	2	0	$\cdots$	0	1	1	0
$\psi_7(x) = 0$	k_7	$\cdots$	7	$\cdots$	2	1	$\cdots$	0	1	1	1
$\psi_8(x) = 0$	k_8	$\cdots$	8	$\cdots$	2	2	$\cdots$	1	0	0	0

By equation (9), we can infer the upper bounds for both bases as:

$$(\mathfrak{d}_{Q'_{\mathcal{B}}})_3 = \max_{j \in B} \left\{ \deg(C_j) \right\} \times |\mathbb{G}| + 4 \times \overline{\deg\left(s_j\right)_3} - |\mathbb{G}_{\mathcal{B}}| \ ,$$

$$(\mathfrak{d}_{Q'_{\mathcal{B}}})_2 = \max_{j \in B} \left\{ \deg(C_j) \right\} \times |\mathbb{G}| + 4 \times \overline{\deg(s_j)_2} - |\mathbb{G}_{\mathcal{B}}| \ .$$

This illustrates that base 2 might not yield the lowest degree bound if $\overline{\deg(s_j)_2} > \overline{\deg(s_j)_3}$. However, the four base 2 selector columns could encode 7 additional constraints, while the two base 3 columns have been entirely used.