An optimization of the addition gate count in Plonkish circuits

Steve Thakur

Panther Protocol

Abstract

We slightly generalize Plonk's ([GWC19]) permutation argument by replacing permutations with arbitrary self-maps. We then use this succinct argument to obtain a protocol for weighted sums on committed vectors, which, in turn, allows us to eliminate the intermediate gates arising from high fan-in additions in Plonkish arithmetic circuits.

We use the KZG polynomial commitment scheme ([KZG10]), which allows for a universal updateable CRS linear in the circuit size. In keeping with our recent work ([Th23]), we have used the monomial basis since it is compatible with any sufficiently large prime scalar field. In settings where the scalar field has a suitable smooth order subgroup, the techniques can be efficiently ported to a Lagrange basis.

The proof size is constant, as is the verification time, which is dominated by a single pairing check (i.e. two pairings). For committed vectors of length n, the proof generation is $O(n \cdot \log(n))$ and is dominated by the $\mathbb{G}_1$ multi-scalar multiplications and a single sum of a few polynomial products over the prime scalar field via multimodular FFTs.[1]

When this weighted sum protocol is merged with the monomial basis Snark described in [Th23], it entails five additional $\mathbb{G}_1$-elements in the proof and thus, adds five $\mathbb{G}_1$-MSMs to the proof generation. It adds a few $\mathbb{G}_1$ scalar multiplications but no additional pairings to the verification. When the analogous protocol in the Lagrange basis is merged with Plonk ([GWC19]), the added costs are similar.

1 Introduction

We generalize the permutation argument in Plonkish arithmetization to arbitrary (possibly non-injective) self-maps of an interval $[0,\ N-1]$. More precisely, for index sets $\mathcal{I},\ \mathcal{J}\ \subseteq\ [0,\ N-1]$ and a committed map[2] $\boldsymbol{\rho} : \mathcal{I} \longrightarrow \mathcal{J}$ with domain $\mathcal{I}$ and image $\mathcal{J}$, we describe a protocol to succinctly show that two committed vectors $\mathbb{V}$, $\widetilde{\mathbb{V}}$ in $\mathbb{F}_p^N$ are linked by the map $\boldsymbol{\rho}$ as follows:

$$(1.1) \qquad \widetilde{\mathbb{V}}[\boldsymbol{\rho}(i)] \ = \ \mathbb{V}[i] \quad \forall\, i \ \in \ \mathcal{I}.$$

The protocol does not assume $\boldsymbol{\rho}$ to be injective and in this sense, is a minor generalization of the permutation argument that is pivotal to Plonkish arithmetization.

As an application of this succinct argument, we then describe a protocol to show that for committed vectors $\mathbb{V}$, $\widetilde{\mathbb{V}} \ \in \ \mathbb{F}_p^N$ and another committed vector $\mathbb{W} \in \mathbb{F}_p^N$ (which will function as the vector of weights), all of the following equations hold:

$$(1.2) \qquad \widetilde{\mathbb{V}}[j] \ = \sum_{i \in \boldsymbol{\rho}^{-1}(j)} \mathbb{W}[i] \cdot \mathbb{V}[i] \quad \forall\, j \ \in \ \mathcal{J},$$

[1]The Prover uses ordinary FFTs in settings where the scalar field has high 2-adicity

[2]see subsection 1.7 for the (straightforward) definition of this commitment

where $\boldsymbol{\rho}^{-1}(j)$ denotes the pre-image set $\{i \in \mathcal{I} : \ \boldsymbol{\rho}(i) = j\}$. An inherent limitation of this protocol is that each index $i \in [0,\ N-1]$ appears in at most one linear relation and hence, the protocol does not quite make addition gates "free" as in [Groth16]. But it allows for the elimination of intermediate gates arising from high fan-in linear relations in the circuit.

We use the KZG polynomial commitment scheme instantiated with a pairing friendly elliptic curve. Furthermore, we have opted for the monomial basis since it happens to be better suited to our use cases[3] and is compatible with arbitrary prime scalar fields of a suitable size. However, these techniques can be easily ported to a setting with a Lagrange basis, assuming the scalar field of the elliptic curve has a large enough subgroup of smooth order. In fact, when the field has a large smooth order subgroup, the monomial basis can be succinctly linked to the Lagrange basis as described in the blogpost [Bl22] by Remco Bloemen.

The proof is constant-sized and can be efficiently merged with the the Snark described in [Th23]. When merged with this scheme, this protocol for weighted sums entails five additional $\mathbb{G}_1$-elements in the proof and thus, five additional $\mathbb{G}_1$ MSMs in the proof generation. The Verifier requires no pairings in addition to the two pairings in the Snark, but does require a few more scalar multiplications in $\mathbb{G}_1$. When the analogous protocol in the Lagrange basis with a smooth order subgroup of $\mathbb{F}_p^*$ is merged with Plonk ([GWC19]), the added costs are similar.

The recent scheme cqLin ([EG23]) achieves the goal of making addition gates effectively free and has a linear Prover time. But it requires a quadratic sized CRS and $O(n^2 \cdot \log(n))$ preprocessing time, which makes it expensive for larger circuits. Furthermore, this scheme hinges on the elegant Feist-Khovratovich trick ([FK]) for computing all KZG opening proofs over a subgroup H of $\mathbb{F}_p^*$ in runtime $O(|H| \cdot \log(|H|))$. As far as we know, the Feist-Khovratovich trick is not easily adaptable to settings where $\mathbb{F}_p^*$ lacks large smooth order subgroups.

1.1 Brief overview of the trick used

Equation 1.1 boils down to showing that for a randomly and uniformly generated challenge $\delta \in \mathbb{F}_p$, the vectors

$$\left[\mathbb{V}[i] + \delta \cdot \boldsymbol{\rho}(i) \ : \ i \in \mathcal{I}\right] \quad \text{and} \quad \left[\widetilde{\mathbb{V}}[j] + \delta \cdot j \ : \ j \in \mathcal{J}\right]$$

have the same underlying set and each $\widetilde{\mathbb{V}}[j] + \delta \cdot j$ occurs in the first vector with multiplicity $\left|\boldsymbol{\rho}^{-1}(j)\right|$, where $\boldsymbol{\rho}^{-1}(j)$ is the pre-image $\{i \in \mathcal{I} : \ \boldsymbol{\rho}(i) = j\}$.

The Schwartz-Zippel lemma implies that Equation 1.2 reduces to proving that the equation

$$\sum_{j \in \mathcal{J}} \widetilde{\mathbb{V}}[j] \cdot \zeta^j \ = \sum_{i \in \boldsymbol{\rho}^{-1}(j)} \mathbb{W}[i] \cdot \mathbb{V}[i] \cdot \zeta^{\boldsymbol{\rho}(i)} \tag{1.3}$$

holds for some randomly generated challenge $\zeta \in \mathbb{F}_p$. The left hand side of equation 1.3 is the dot product $\widetilde{\mathbb{V}} \circ \boldsymbol{\chi}_{\mathcal{J},\zeta}$, where $\boldsymbol{\chi}_{\mathcal{J},\zeta} \in \mathbb{F}_p^{\max(\mathcal{J})+1}$ is the twist by ζ of the "indicator vector" of $\mathcal{J}$[4], i.e. the vector given by

$$\boldsymbol{\chi}_{\mathcal{J},\zeta}[k] = \begin{cases} \zeta^k \text{ if } k \in \mathcal{J} \\ 0 \text{ if } k \notin \mathcal{J} \end{cases}$$

The right hand side of equation of 1.3 is $[\mathbb{W} \odot \mathbb{V}] \circ \mathbb{S}_{\boldsymbol{\rho},\zeta}$, where $\odot$, $\circ$ denote the Hadamard (aka

[3]In particular, outer curves to Ed25519 and BN254

[4]In the monomial basis, the corresponding polynomials are given by $\chi_{\mathcal{J}}(X) := \sum_{j \in \mathcal{J}} X^j,\ \chi_{\mathcal{J},\zeta}(X) := \chi_{\mathcal{J}}(\zeta \cdot X)$

entrywise) and dot products respectively and $\mathbb{S}_{\boldsymbol{\rho},\zeta}$ denotes the vector given by

$$\mathbb{S}_{\boldsymbol{\rho},\zeta}[k] = \begin{cases} \zeta^{\boldsymbol{\rho}(k)} \text{ if } k \in \mathcal{I} \\ 0 \text{ if } k \notin \mathcal{I} \end{cases}$$

Thus, aside from the protocols for the Hadamard and dot products, equation 1.3 boils down to verifiably sending a commitment to the vector $\mathbb{S}_{\boldsymbol{\rho},\zeta}$. This is the unique vector of length $\leq N$ that satisfies the relation

$$\mathbb{S}_{\boldsymbol{\rho},\zeta}[i] \;=\; \boldsymbol{\chi}_{\mathcal{J},\zeta}[\boldsymbol{\rho}(i)] \quad \forall\, i \;\in\; \mathcal{I} \tag{1.4}$$

to the vector $\boldsymbol{\chi}_{\mathcal{J},\zeta}$ and has entries 0 at all positions outside the index set $\mathcal{I}$. Thus, equation 1.3 (and hence, equation 1.2) boils down to succinctly proving an upper bound on the length of the committed vector $\mathbb{S}_{\boldsymbol{\rho},\zeta}$ and showing that the map $\boldsymbol{\rho}$ links $\mathbb{S}_{\boldsymbol{\rho},\zeta}$ to $\boldsymbol{\chi}_{\mathcal{J},\zeta}$ in the sense of equation 1.1.

1.2 The setup

Let $\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T$ be cyclic groups of order p for some prime p such that there exists a pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ which is *bilinear*, *non-degenerate* and *efficiently computable*. We fix generators $\mathbf{g}_1,\ \mathbf{g}_2$ in $\mathbb{G}_1,\ \mathbb{G}_2$ respectively. For a trapdoor $\mathbf{s} \in \mathbb{F}_p^*$, the common reference string (CRS) generated via a multi-party computation is given by

$$[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ \ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$$

for an appropriate upper bound M. The verification key is $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}}],\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$.

We define a simple vector commitment using the KZG polynomial commitment scheme. A vector $\mathbf{v} = (v_0, \cdots, v_{n-1}) \in \mathbb{F}_p^n$ is identified with the polynomial $\sum_{i=0}^{n-1} v_i \cdot X^i$, which is then committed as in [KZG10]. Thus, for a vector $\mathbf{v} = (v_0, \cdots, v_n) \in \mathbb{F}_p^{n+1}$, we define the commitment

$$\texttt{Com}(\mathbf{v}) \;:=\; \mathbf{g}_1^{\sum\limits_{i=0}^{n} v_i \cdot \mathbf{s}^i} \;=\; \prod_{i=0}^{n} (\mathbf{g}_1^{\mathbf{s}^i})^{v_i} \;\in\; \mathbb{G}_1.$$

1.3 Notations and terminology

As usual, $\mathbb{F}_q$ denotes the finite field with q elements for a prime power q and $\overline{\mathbb{F}}_q$ denotes its algebraic closure. $\mathbb{F}_q^*$ denotes the cyclic multiplicative group of the non-zero elements of $\mathbb{F}_q$. $\mathbb{F}_q[X]$ denotes the ring of univariate polynomials over $\mathbb{F}_q$, which is a principal ideal domain. $\mathbb{F}_q(X)$ denotes the field $\mathrm{Frac}(\mathbb{F}_q[X])$, the fraction field of $\mathbb{F}_q[X]$.

For a polynomial $f(X)$, $\deg(f)$ denotes its degree and $\texttt{Coef}(f\ ,\ i)$ denotes the coefficient at the position X^i. $f'(X)$ denotes the derivative of $f(X)$.

We fix a hashing algorithm $\texttt{Hash}_{\text{FS}}$ that generates random and uniform challenges in $\mathbb{F}_p$ to make the protocols non-interactive.

We denote by $\lambda_{\texttt{sec}} \in \mathbb{Z}^+$ a security parameter. We denote by $\texttt{negl}(\lambda_{\texttt{sec}})$ an unspecified function that is *negligible* in $\lambda_{\texttt{sec}}$ (namely, a function that vanishes faster than the inverse of any polynomial in $\lambda_{\texttt{sec}}$). When a function can be expressed in the form $1 - \texttt{negl}(\lambda_{\texttt{sec}})$, we say that it is *overwhelming* in $\lambda_{\texttt{sec}}$. We say some events are equivalent with overwhelming probability, if the probability of any proper subset of this set of events being true and the other events false is negligible in $\lambda_{\texttt{sec}}$.

Definition 1.1. An argument system is *complete* if an honest Prover can efficiently output an accepting transcript.

Definition 1.2. An argument system is *sound* if the probability of a cheating Prover successfully convincing a Verifier is negligible.

Definition 1.3. An argument system is *knowledge sound* if for any probabilistic polynomial time algorithm $\mathcal{A}_{\texttt{PPT}}$ that outputs an accepting transcript, there exists an extractor $\mathcal{E}_{\texttt{PPT}}$ that, with overwhelming probability, succeeds in extracting a valid witness.

1.4 Hardness assumptions

We state the computationally infeasible problems that the security of our constructions hinges on.

Assumption 1.1. n-strong Diffie Hellman assumption: *Let $\mathbb{G}$ be a cyclic group of prime order p generated by an element $\mathbf{g}$, and let $\mathbf{s} \in \mathbb{F}_p^*$. Any probabilistic polynomial-time algorithm that is given the set $\{\mathbf{g}^{\mathbf{s}^i} : 1 \le i \le n\}$ can output a pair $(\alpha,\ \mathbf{g}^{1/(\mathbf{s}+\alpha)}) \in \mathbb{F}_p^* \times \mathbb{G}$ with at most negligible probability.*

Assumption 1.2. Knowledge of exponent assumption (KEA): *Let $\mathbb{G}$ be a cyclic group of prime order p generated by an element $\mathbf{g}$, and let $\mathbf{s} \in \mathbb{F}_p^*$. Suppose there exists a PPT algorithm $\mathcal{A}_1$ that given pairs $(h_1, h_1^{\mathbf{s}}), \cdots, (h_n, h_n^{\mathbf{s}})$ in $\mathbb{G}^2$, outputs a pair $(\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{G}^2$ such that $\mathbf{C}_2 = \mathbf{C}_1^{\mathbf{s}}$. Then there exists a PPT algorithm $\mathcal{A}_2$ that, with overwhelming probability, outputs a vector $(x_1, \cdots, x_n) \in \mathbb{F}_p^n$ such that*

$$\mathbf{C}_1 = \prod_{i=1}^{n} h_i^{x_i} \quad , \quad \mathbf{C}_2 = \prod_{i=1}^{n} (h_i^{\mathbf{s}})^{x_i}$$

A special case of the KEA assumption is that given the elements $\{\mathbf{g}^{\mathbf{s}^i} : 0 \le i \le n\}$, if a PPT algorithm $\mathcal{A}_1$ is able to output a triplet $(\mathbf{C}_1, \mathbf{C}_2, f(X)) \in \mathbb{G} \times \mathbb{G} \times \mathbb{F}_p[X]$ with $\deg(f(X) \ge 1$ such that $\mathbf{C}_2 = \mathbf{C}_1^{f(\mathbf{s})}$, then there is a PPT algorithm $\mathcal{A}_2$ that with overwhelming probability, outputs a polynomial $e(X)$ such that

$$\mathbf{C}_1 = \mathbf{g}^{e(\mathbf{s})} \quad , \quad \mathbf{C}_2 = \mathbf{g}^{e(\mathbf{s}) \cdot f(\mathbf{s})}.$$

1.5 The AGM model

In order to achieve additional efficiency, we also construct polynomial commitment schemes in the Algebraic Group Model (AGM) [FKL18], which replaces specific knowledge assumptions (such as Power Knowledge of Exponent assumptions). In our protocols, by an algebraic adversary $\mathcal{A}_{\texttt{PPT}}$ in a CRS-based protocol, we mean a PPT algorithm which satisfies the following:

Whenever $\mathcal{A}_{\texttt{PPT}}$ outputs an element $\mathbf{A} \in \mathbb{G}_i$ $(i = 1, 2)$, it also outputs a vector $\mathbf{v} = (v_0, \cdots, v_{n-1}) \in \mathbb{F}_p^n$ such that

$$\mathbf{A} \;=\; \big\langle \mathbf{v}\ ,\ \texttt{CRS} \big\rangle \;=\; \prod_{i=0}^{n-1} (\mathbf{g}_1^{\mathbf{s}^n})^{v_i} \;=\; \mathbf{g}_1^{\sum_{i=0}^{n-1} v_i \cdot \mathbf{s}^i} .$$

The AGM allows a Prover to commit to multiple polynomials $f_i(X) \in \mathbb{F}_p[X]$ of a bounded degree and open these polynomials at some point $\alpha \in \mathbb{F}_p$. To show that $f_i(\alpha) = \beta_i$ for each index i , it suffices for the Prover to show that for a randomly and uniformly generated challenge λ, the polynomial

$$f_\lambda(X) \;:=\; \sum_i \lambda^{i-1} \cdot f_i(X)$$

is valued $\beta := \sum_i \lambda^{i-1} \cdot \beta_i$ at $X = \alpha$. If the Prover were dishonest about one or more of the elements $f(\alpha_i)$, the pairing check would fail with overwhelming probability.

The algebraic group model implies that there is an efficient extractor $\mathcal{E}_{\mathtt{multi-PC}}$ that - given access to the multi-commitment opening proof - can extract the polynomials in expected polynomial time. We refer the reader to [GWC19], [CHHMVW20] and [FKL18] for a more detailed exposition of the AGM.

1.6 Commitments to index sets

For an index set $I \subseteq [0,\ \text{length}(\mathtt{CRS})]$, we commit to the set $\mathcal{I}$ by committing to the polynomial

$$\chi_{\mathcal{I}}(X) \ := \ \sum_{i \in \mathcal{I}} X^i,$$

which we refer to as the *indicator polynomial* of $\mathcal{I}$. Thus, the commitment is given by

$$\mathtt{Com}(\mathcal{I}) \ := \ \mathtt{Com}(\chi_{\mathcal{I}}(X)) \ = \ \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})} \ = \ \mathbf{g}_1^{\sum\limits_{i \in \mathcal{I}} \mathbf{s}^i}$$

The polynomial $\chi_{\mathcal{I}}(X)$ is binary in the sense that every coefficient lies in $\{0,1\} \subseteq \mathbb{F}_p$. Conversely, every binary polynomial of degree $\leq n$ is of the form $\chi_{\mathcal{I}}(X)$ for some subset $I \subseteq [0,\ n]$.

1.7 Commitments to permutations and self-maps of $[0,\ N-1]$

For a permutation $\boldsymbol{\sigma} :\ [0,\ N-1] \longrightarrow [0,\ N-1]$, we commit to $\boldsymbol{\sigma}$ by committing to the polynomial

$$S_{\boldsymbol{\sigma}}(X) := \sum_{i=0}^{N-1} \boldsymbol{\sigma}(i) \cdot X^i.$$

In particular, we commit to the identity permutation of $[0,\ N-1]$ by committing to the polynomial $P_{\mathtt{id},N}(X) := \sum_{i=0}^{N-1} k \cdot X^k$.

Similarly, for a (possibly non-injective) map $\boldsymbol{\rho} :\ \mathcal{I} \longrightarrow \mathcal{J}$ of index sets $\subseteq\ [0,\ N-1]$ with domain $\mathcal{I}$ and image $\mathcal{J}$, we commit to $\boldsymbol{\rho}$ by committing to the polynomial

$$S_{\boldsymbol{\rho}}(X) := \sum_{i=0}^{N-1} \boldsymbol{\rho}(i) \cdot X^i$$

and to the indicator polynomial $\chi_{\mathcal{I}}(X)$ of $\mathcal{I}$. Thus, this commitment consists of two $\mathbb{G}_1$ elements.

1.8 The Hadamard product

For polynomials $f_1(X)$, $f_2(X)$, the *Hadamard product* $f_1 \odot f_2(X)$ (or $f_1(X) \odot f_2(X)$) is given by

$$f_1 \odot f_2(X) \ := \ \sum_{i=0}^{\min(\deg(f_1)\ ,\ \deg(f_2))} \mathtt{Coef}(f_1\ ,\ i) \cdot \mathtt{Coef}(f_2\ ,\ i) \cdot X^i.$$

For instance, for an index set $\mathcal{I}$ with indicator polynomial $\chi_{\mathcal{I}}(X)$, we have

$$f(X) \odot \chi_{\mathcal{I}}(X) \ = \ \sum_{i \in \mathcal{I}} \mathtt{Coef}(f\ ,\ i) \cdot X^i.$$

The *dot product* $f_1 \circ f_2(X)$ is the evaluation of the Hadamard product $f_1 \odot f_2(X)$ at $X = 1$.

For a fixed integer $N \geq \deg(f_2)$ and a randomly generated challenge γ, the product

$$f_\gamma(X) \;:=\; f_1(\gamma \cdot X) \cdot X^N \cdot f_2(X^{-1})$$

is a polynomial of degree

$$\deg(f_\gamma) \;=\; \deg(f_1) + N - \mathrm{val}_{(X)}(f_2(X)) \;\leq\; \deg(f_1) + N.$$

Its coefficient $\texttt{Coef}(f_\gamma\ ,\ N)$ at X^N is given by the sum

$$\sum_{i=0}^{\min(\deg(f_1)\ ,\ \deg(f_2))} \texttt{Coef}(f_1\ ,\ i) \cdot \texttt{Coef}(f_2\ ,\ i) \cdot \gamma^i\ \ ,$$

which happens to coincide with the evaluation of the Hadamard product $f_1 \odot f_2(X)$ at γ. We exploit this simple fact in conjunction with the protocol for the degree upper bound to obtain a protocol for the Hadamard product.

Showing that a committed polynomial is divisible by the monomial X^{N+1} is straightforward. To show that a committed polynomial $f(X)$ is of degree $\leq n$ for a public integer n, the Prover verifiably sends a commitment to the polynomial $\widehat{f}(X) := X^n \cdot f(X^{-1})$. This implies that with overwhelming probability, the rational function $X^n \cdot f(X^{-1})$ is a polynomial, whence it follows that $\deg(f) \leq n$.

1.9 The degree upper bound

We describe the subprotocol that shows that for a committed polynomial $f(X)$ and a public integer n, we have the degree upper bound $\deg(f) \leq n$. It hinges on the simple observation that

$$\deg(f) \leq n \iff X^n \cdot f(X^{-1}) \;\in\; \mathbb{F}_p[X].$$

Thus, a Prover can demonstrate this upper bound on the degree by verifiably sending the KZG commitment to the polynomial $\widehat{f}(X) := X^n \cdot f(X^{-1})$. This can be accomplished by showing that for a random challenge α, the equality $\widehat{f}(\alpha^{-1}) = \alpha^{-n} \cdot f(\alpha)$ holds.

We note that the protocol is batchable. For committed polynomials $f_i(X)$ and integers n_i, we have $\deg(f_i) \leq n_i$ for each index i if and only if, for a randomly generated challenge λ, the rational function

$$f_\lambda(X) \;:=\; \sum_{i=1}^{k} \lambda^{i-1} \cdot X^{n_i} \cdot f_i(X^{-1})$$

is a polynomial (lemma 1.1). Thus, a Prover can demonstrate all of these degree upper bounds by verifiably sending the KZG commitment to $f_\lambda(X)$.

1.10 Preliminary lemmas

We will need the following elementary lemmas.

Lemma 1.1. *For rational functions $h_i(X) \in \mathbb{F}_p(X) := \mathrm{Frac}(\mathbb{F}_p[X])$, if the sum $\sum_{i=1}^{k} \lambda^{i-1} \cdot h_i(X)$ is a polynomial for a randomly generated $\lambda \in \mathbb{F}_p$, then with overwhelming probability, each rational function $h_i(X)$ is a polynomial.*

Proof. Suppose there exists at least one index j such that $h_j(X)$ is not a polynomial. Let $q(X) \in \mathbb{F}_p[X]$ be an irreducible polynomial such that $\mathrm{val}_{q(X)}(h_j(X)) \leq -1$, i.e. $h_i(X) = h_{i,1}(X)/h_{i,2}(X)$ with $h_{i,1}(X),\ h_{i,2}(X) \in \mathbb{F}_p[X]$ co-prime and $h_{i,2}(X)$ divisible by $q(X)$.

Set $f_i(X) = q(X) \cdot h_i(X)$ for $i = 1, \cdots, k$. Then

$$\sum_{i=1}^{k} \lambda^{i-1} \cdot h_i(X) = q(X)^{-1} \cdot \Big[\sum_{i=1}^{k} \lambda^{i-1} \cdot f_i(X) \Big] \;\in\; \mathbb{F}_p[X]$$

and hence, $\sum_{i=1}^{k} \lambda^{i-1} \cdot f_i(X)$ is divisible by $q(X)$. Applying the Schwartz-Zippel lemma to the quotient field $\mathbb{F}_p[X]/(q(X))$ implies that with overwhelming probability, $q(X)$ divides each $f_i(X)$, a contradiction. □

The permutation argument from [GWC19] exploits the fact that for a permutation $\boldsymbol{\sigma}$ of $[0,\ N-1]$ and vectors $\mathbb{V}$, $\widetilde{\mathbb{V}}$, the following are equivalent with overwhelming probability:

1. $\boldsymbol{\sigma}(\mathbb{V}) \;=\; \widetilde{\mathbb{V}}$
2. For a randomly generated element $\delta_1 \in \mathbb{F}_p$, the vectors $\big[\mathbb{V}[i] + \delta_1 \cdot \boldsymbol{\sigma}(i) \;:\; i \in [0,\ N-1]\big]$ and $\big[\widetilde{\mathbb{V}}[j] + \delta_1 \cdot j \;:\; j \in [0,\ N-1]\big]$ have the same multiset.
3. the equation

$$\prod_{i=0}^{N-1} (X + \mathbb{V}[i] + \delta_1 \cdot \boldsymbol{\sigma}(i)) \;\;=\;\; \prod_{j=0}^{N-1} (X + \widetilde{\mathbb{V}}[j] + \delta_1 \cdot j) \;\;\in\;\; \mathbb{F}_p[X]$$

holds for a randomly generated challenge δ_1.

4. the equation

$$\prod_{i=0}^{N-1} (\mathbb{V}[i] + \delta_1 \cdot \boldsymbol{\sigma}(i) + \delta_2) \;\;=\;\; \prod_{j=0}^{N-1} (\widetilde{\mathbb{V}}[j] + \delta_1 \cdot j + \delta_2) \;\;\in\;\; \mathbb{F}_p$$

holds for randomly generated challenges $\delta_1,\ \delta_2$.

The next lemma yields a generalization of the permutation argument to non-injective maps $\boldsymbol{\rho} :\ \mathcal{I} \;\longrightarrow\; \mathcal{J}$ of subsets of an interval $[0,\ N-1]$. The basic idea is to fixate on the underlying sets of these vectors rather than the multisets.

Lemma 1.2. *Let $\mathcal{I}$, $\mathcal{J}$ be subsets of the interval $[0,\ N-1]$ and let $\boldsymbol{\rho} :\ \mathcal{I} \;\longrightarrow\; \mathcal{J}$ be a map with domain $\mathcal{I}$ and image $\mathcal{J}$.*

For vectors $\mathbb{V}$, $\widetilde{\mathbb{V}}$ of $\mathbb{F}_p$-elements, the following statements are equivalent with overwhelming probability:

(1). $\widetilde{\mathbb{V}}[\boldsymbol{\rho}(i)] \;=\; \mathbb{V}[i] \qquad \forall\, i \in \mathcal{I}$.

(2). *For a randomly generated element $\delta \in \mathbb{F}_p$, the vectors*

$$\big[\mathbb{V}[i] + \delta \cdot \boldsymbol{\rho}(i) \;:\; i \in \mathcal{I}\big] \quad \textit{and} \quad \big[\widetilde{\mathbb{V}}[j] + \delta \cdot j \;:\; j \in \mathcal{J}\big]$$

have the same underlying set.

(3). *There exists a sequence $m_j \in \mathbb{F}_p$ $(j \in \mathcal{J})$ such that for a randomly generated element $\delta \in \mathbb{F}_p$, the equation*

$$\sum_{i \in \mathcal{I}} \big(X + \mathbb{V}[i] + \delta \cdot \boldsymbol{\rho}(i)\big)^{-1} \;\;=\;\; \sum_{j \in \mathcal{J}} m_j \cdot \big(X + \widetilde{\mathbb{V}}[j] + \delta \cdot j\big)^{-1}$$

of rational functions holds.

(4). *There exists a sequence $m_j \in \mathbb{F}_p$ $(j \in \mathcal{J})$ such that for randomly generated elements $\delta, \alpha \in \mathbb{F}_p$, the equation*

$$\sum_{i\in\mathcal{I}} \big(\alpha + \mathbb{V}[i] + \delta\cdot\boldsymbol{\rho}(i)\big)^{-1} \;=\; \sum_{j\in\mathcal{J}} m_j \cdot \big(\alpha + \widetilde{\mathbb{V}}[j] + \delta\cdot j\big)^{-1} \;\in\; \mathbb{F}_p$$

holds.

Proof. (1) $\iff$ (2) is straightforward. (3) $\iff$ (4) is an immediate implication of the Schwartz-Zippel lemma.

(2) $\iff$ (3) is well-known and has been used in [Hab22], [EFG22] etc. It hinges on the observation that defining the polynomials

$$f_{\Pi,\mathbb{V},\boldsymbol{\rho},\mathcal{I}}(X) \;:=\; \prod_{i\in\mathcal{I}} X + \mathbb{V}[i] + \lambda\cdot\boldsymbol{\rho}(i) \quad,\quad f_{\Pi,\widetilde{\mathbb{V}},\boldsymbol{\rho},\mathcal{J}}(X) \;:=\; \prod_{j\in\mathcal{J}} \big(X + \widetilde{\mathbb{V}}[j] + \lambda\cdot j\big)^{\mathrm{mul}(j,\boldsymbol{\rho})}$$

yields

$$\sum_{i\in\mathcal{I}} \big(X + \mathbb{V}[i] + \lambda\cdot\boldsymbol{\rho}(i)\big)^{-1} = \frac{f'_{\Pi,\mathbb{V}_1,\boldsymbol{\rho},\mathcal{I}}(X)}{f_{\Pi,\mathbb{V},\boldsymbol{\rho},\mathcal{I}}(X)} \quad,\quad \sum_{j\in\mathcal{J}} \mathrm{mul}(j,\boldsymbol{\rho})\cdot\big(X + \widetilde{\mathbb{V}}[j] + \lambda\cdot j\big)^{-1} = \frac{f'_{\Pi,\widetilde{\mathbb{V}},\mathcal{J}}(X)}{f_{\Pi,\widetilde{\mathbb{V}},\boldsymbol{\rho},\mathcal{J}}(X)},$$

where the polynomials $f'_{\Pi,\mathbb{V},\boldsymbol{\rho},\mathcal{I}}(X)$, $f'_{\Pi,\widetilde{\mathbb{V}},\boldsymbol{\rho},\mathcal{J}}(X)$ are the derivatives of the polynomials $f_{\Pi,\mathbb{V},\boldsymbol{\rho},\mathcal{I}}(X)$, $f_{\Pi,\widetilde{\mathbb{V}},\boldsymbol{\rho},\mathcal{J}}(X)$ respectively and $\mathrm{mul}(j,\boldsymbol{\rho})$ denotes the multiplicity of j with respect to $\boldsymbol{\rho}$, i.e. the cardinality of the pre-image set $\boldsymbol{\rho}^{-1}(j) \;:=\; \{i \in \mathcal{I} :\; \boldsymbol{\rho}(i) = j\}$.

For *monic* polynomials $h_1(X)$, $h_2(X)$, we have

$$\frac{h_1'(X)}{h_1(X)} = \frac{h_2'(X)}{h_2(X)} \implies \left(\frac{h_1(X)}{h_2(X)}\right)' = h_1(X)\cdot h_2'(X) - h_1'(X)\cdot h_2(X) = 0 \implies h_1(X) = h_2(X),$$

which completes the proof of (2), (3) being equivalent with overwhelming probability. □

2 Preliminary subprotocols

In this section, we describe the subprotocols underpinning the main protocols of the paper. We start out with the simple protocol for the twist, which will be necessary for some of the subsequent protocols.

$$\mathcal{R}_{\texttt{Twist}}[\mathbf{g}_1,\ (\mathbf{a},\gamma),\ \mathbf{a}_\gamma] = \{(\mathbf{a},\mathbf{a}_\gamma \in \mathbb{G}_1,\ \gamma\in\mathbb{F}_p),\ f(X)\in\mathbb{F}_p[X]) : \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}\ ,\ \mathbf{g}_1^{f(\gamma\cdot\mathbf{s})} = \mathbf{a}_\gamma\}$$

Protocol 2.1. *Proof of twist* (`PoTwist`):

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $\mathbf{g}_1,\ \mathbf{g}_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively. The CRS $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]$, $[\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

Common Inputs: Elements $\mathbf{a},\ \mathbf{a}_\gamma \in \mathbb{G}_1$; element $\gamma \in \mathbb{F}_p$.

Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})} \ ,\ \mathbf{a}_\gamma = \mathbf{g}_1^{f(\gamma\cdot\mathbf{s})}.$$

1. The hashing algorithm $\texttt{Hash}_{\texttt{FS}}$ generates a challenge $\alpha \in \mathbb{F}_p^*$.

2. The Prover sends the $\mathbb{F}_p$-element $\beta := f(\alpha)$ and the $\mathbb{G}_1$-elements

$$\mathbf{Q} := \mathbf{g}_1^{[f(\mathbf{s})-\beta]\big/[\mathbf{s}-\alpha]} \;\;,\;\; \mathbf{Q}_\gamma := \mathbf{g}_1^{[f(\gamma\cdot\mathbf{s})-\beta]\big/[\gamma\cdot\mathbf{s}-\alpha]}.$$

3. The Verifier $\mathcal{V}$ verifies the (batchable) equations

$$\mathbf{e}(\mathbf{Q}\,,\,\mathbf{g}_2^{\mathbf{s}-\alpha}) \overset{?}{=} \mathbf{e}(\mathbf{a}\cdot\mathbf{g}_1^{-\beta}\,,\,\mathbf{g}_2) \;\;,\;\; \mathbf{e}(\mathbf{Q}_\gamma\,,\,\mathbf{g}_2^{\gamma\cdot\mathbf{s}-\alpha}) \overset{?}{=} \mathbf{e}(\mathbf{a}_1\cdot\mathbf{g}_1^{-\beta}\,,\,\mathbf{g}_2).$$

□

2.1 Protocol for the degree upper bound

$$\mathcal{R}_{\texttt{DegUp}}[\mathbf{g}_1,\ (\mathbf{a},\ n)] = \{(\mathbf{a}\in\mathbb{G}_1,\ n\in\mathbb{Z}),\ f(X)\in\mathbb{F}_p[X]) : \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}\ ,\ \deg(f)\le n\}$$

Protocol 2.2. *Proof of degree upper bound* (`PoDegUp`):

Parameters: A pairing $\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\longrightarrow\mathbb{G}_T$; generators $\mathbf{g}_1,\ \mathbf{g}_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively. The CRS $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

Common Inputs: Elements $\mathbf{a}\in\mathbb{G}_1$, $n\in\mathbb{Z}$.

Claim: The Prover knows a polynomial $f(X)\in\mathbb{F}_p[X]$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})} \;\;,\;\; \deg(f)\le n.$$

1. The Prover $\mathcal{P}$ computes $\widehat{f}(X) := X^n\cdot f(X^{-1})$ and sends the $\mathbb{G}_1$-element $\widehat{\mathbf{a}} := \mathbf{g}_1^{\widehat{f}(\mathbf{s})}$.
2. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\alpha\in\mathbb{F}_p^*$.
3. The Prover computes the polynomials $q(X)$, $\widehat{q}(X)$ such that

$$f(X) = q(X)\cdot(X-\alpha)+f(\alpha) \;\;,\;\; \widehat{f}(X) = \widehat{q}(X)\cdot(X-\alpha^{-1})+\alpha^{-n}\cdot f(\alpha)$$

and sends the $\mathbb{G}_1$-elements

$$\mathbf{Q} := \mathbf{g}_1^{q(\mathbf{s})} \;\;,\;\; \widehat{\mathbf{Q}} := \mathbf{g}_1^{\widehat{q}(\mathbf{s})}$$

and the $\mathbb{F}_p$-element $\beta := f(\alpha)$.

4. The Verifier $\mathcal{V}$ computes $\widehat{\beta} := \alpha^{-n}\cdot\beta$ and verifies the equations

$$\mathbf{Q}^{\mathbf{s}-\alpha} \overset{?}{=} \mathbf{a}\cdot\mathbf{g}_1^{-\beta} \;\;,\;\; \widehat{\mathbf{Q}}^{\mathbf{s}-\alpha^{-1}} \overset{?}{=} \widehat{\mathbf{a}}\cdot\mathbf{g}_1^{-\widehat{\beta}}$$

via the (batchable) pairing checks

$$\mathbf{e}(\mathbf{Q}\,,\,\mathbf{g}_2^{\mathbf{s}-\alpha}) \overset{?}{=} \mathbf{e}(\mathbf{a}\cdot\mathbf{g}_1^{-\beta}\,,\,\mathbf{g}_2) \;\;,\;\; \mathbf{e}(\widehat{\mathbf{Q}}\,,\,\mathbf{g}_2^{\mathbf{s}-\alpha^{-1}}) \overset{?}{=} \mathbf{e}(\widehat{\mathbf{a}}\cdot\mathbf{g}_1^{-\widehat{\beta}}\,,\,\mathbf{g}_2).$$

□

Proposition 2.3. *The protocol* `PoDegUp` *is secure in the algebraic group model.*

Proof. Appendix of [Th23]. □

2.2 Batched proof of divisibility

For committed polynomials $h_i(X)$ $(i = 1, \cdots, k)$ and publicly known sparse polynomials $e_i(X)$, we describe a protocol to show that $e_i(X)$ divides $h_i(X)$ for each index i. The proof consists of 2 $\mathbb{G}_1$ elements and k $\mathbb{F}_p$ elements. The goal is to keep the proof size low and to keep the number of MSMs to a bare minimum.

The protocol hinges on the simple observation (lemma 1.1) that for a set of rational functions in $\mathbb{F}_p(X) := \mathrm{Frac}(\mathbb{F}_p[X])$, if a randomized sum of these rational functions is a polynomial, then with overwhelming probability, all of the rational functions are polynomials. The assumption that the $e_i(X)$ are sparse implies that the Verifier can evaluate them at a challenge α.

In particular, for committed polynomials $f_i(X)$ and $\mathbb{F}_p$ elements α_i $(i = 1, \cdots, k)$, setting $h_i(X) := f_i(X) - f(\alpha_i)$, $e_i(X) := X - \alpha_i$ allows us to send a proof of size 2 $\mathbb{G}_1$, k $\mathbb{F}_p$ to open the polynomials $f_i(X)$ at α_i.

Protocol 2.4. *Batched proof of divisibility* (`BatchDiv`)

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $\mathbf{g}_1, \mathbf{g}_2$ for $\mathbb{G}_1, \mathbb{G}_2$ respectively. The CRS $[\mathbf{g}_1, \mathbf{g}_1^{\mathbf{s}}, \cdots, \mathbf{g}_1^{\mathbf{s}^M}]$, $[\mathbf{g}_2, \mathbf{g}_2^{\mathbf{s}}]$

Common Inputs: Elements $\mathbf{b}_i \in \mathbb{G}_1$; sparse public polynomials $e_i(X) \in \mathbb{F}_p$ for indices $i = 1, \cdots, k$

Claim: The Prover knows polynomials $h_i(X)$ such that

$$\mathbf{b}_i = \mathbf{g}_1^{h_i(\mathbf{s})} \ , \ h_i(X) \equiv 0 \pmod{e_i(X)}.$$

1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widetilde{\lambda}$.
2. The Prover $\mathcal{P}$ computes the polynomial

$$h_{\widetilde{\lambda}}(X) \ := \ \sum_{i=1}^{k} \widetilde{\lambda}^{i-1} \cdot e_i(X)^{-1} \cdot h_i(X)$$

and sends the $\mathbb{G}_1$-element

$$\mathbf{B}_{\widetilde{\lambda}} \ := \ \mathbf{g}_1^{h_{\widetilde{\lambda}}(\mathbf{s})}$$

3. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widetilde{\alpha}$.
4. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\beta_i := h_i(\widetilde{\alpha}) \quad (i = 1, \cdots, k)$.
5. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widetilde{\xi}$.
6. $\mathcal{P}$ computes

$$q(X) \ := \ (X - \widetilde{\alpha})^{-1} \cdot \left[[h_{\widetilde{\lambda}}(X) - h_{\widetilde{\lambda}}(\widetilde{\alpha})] \ + \ \sum_{i=1}^{k} \widetilde{\xi}^{i} \cdot [h_i(X) - \beta_i] \right]$$

and sends the $\mathbb{G}_1$-element

$$\widetilde{\mathbf{Q}} \ := \ \mathbf{g}_1^{q(\mathbf{s})}.$$

7. The Verifier $\mathcal{V}$ computes the $\mathbb{F}_p$-element

$$\widetilde{\beta} \ := \ \Big[\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}\Big] \ + \ \sum_{i=1}^{k} \widetilde{\xi}^{i} \cdot \beta_i$$

8. $\mathcal{V}$ verifies the equation

$$\widetilde{\mathbf{Q}}^{\mathbf{s}-\widetilde{\alpha}} \ \overset{?}{=} \ \mathbf{B}_{\widetilde{\lambda}} \cdot \Big[\prod_{i=1}^{k} (\mathbf{b}_i)^{\widetilde{\xi}^i}\Big] \cdot \mathbf{g}_1^{-\widetilde{\beta}}$$

via the pairing check $\mathbf{e}\big(\widetilde{\mathbf{Q}} \ , \ \mathbf{g}_2^{\mathbf{s}-\widetilde{\alpha}}\big) \ \overset{?}{=} \ \mathbf{e}\big(\mathbf{B}_{\widetilde{\lambda}} \cdot \big[\prod_{i=1}^{k} (\mathbf{a}_i)^{\widetilde{\xi}^i}\big] \cdot \mathbf{g}_1^{-\widetilde{\beta}} \ , \ \mathbf{g}_2\big)$. □

Proposition 2.5. *The protocol* `PoBatchDiv` *is secure in the algebraic group model.*

Proof. Appendix of [Th23]. □

2.3 The batched Hadamard product protocol

As mentioned in the introduction, we exploit the fact that the product $f_1(\gamma \cdot X) \cdot X^N \cdot f_2(X^{-1})$ has coefficient $f_1 \odot f_2(\gamma)$ at the position X^N. We note that the protocol is batchable in the sense that to show that:

$$L_j \odot R_j(X) \ = \ O_j(X) \ \text{ for } \ j = 1, \cdots, k \ ,$$

it suffices to show that for randomly generated challenges γ, λ, the sum

$$f_\lambda(X) \ := \ \sum_{j=1}^{k} \lambda^{j-1} \cdot L_j(\gamma \cdot X) \cdot X^N \cdot R_j(X^{-1})$$

has coefficient $\sum_{j=1}^{k} \lambda^{j-1} \cdot O_j(\gamma)$ at the position X^N. This boils down to expressing the difference

$$f_\lambda(X) \ - \ \Big[\sum_{j=1}^{k} \lambda^{j-1} \cdot O_j(\gamma)\Big] \cdot X^N$$

as a sum of a polynomial $f_{\lambda,-}(X)$ of degree $\leq N-1$ and a polynomial $f_{\lambda,+}(X)$ divisible by X^{N+1}.

We compute this sum of polynomial products over the prime scalar field using the multimodular FFT algorithm. This entails one FFT per product and per prime modulus used. Retrieving $f_\lambda(X)$ in the coefficient form requires a single inverse FFT per prime modulus followed by the Chinese remainder theorem. We note that for polynomial products over *prime* finite fields, the multimodular FFT outperforms Schönhage-Strassen ([SS71]) and the ECFFT ([BCKL21]).

$$\mathcal{R}_{\texttt{HadProd}}\big[\mathbf{g}_1, \ (\mathbf{a}_{L,j}, \mathbf{a}_{R,j})_{j=1}^{k}, \ (\mathbf{a}_{O,j})_{j=1}^{k}\big] = \left\{ \begin{array}{l} \big((\mathbf{a}_{L,j}, \ \mathbf{a}_{R,j}, \ \mathbf{a}_{O,j} \in \mathbb{G}_1) \ , \ L_j(X), \ R_j(X) \in \ \mathbb{F}_p[X]\big) : \\ \mathbf{g}_1^{L_j(\mathbf{s})} = \mathbf{a}_{L,j} \ , \ \mathbf{g}_1^{R_j(\mathbf{s})} = \mathbf{a}_{R,j} \ , \ \mathbf{g}_1^{L_j \odot R_j(\mathbf{s})} = \mathbf{a}_{O,j} \end{array} \right\}$$

Protocol 2.6. *Batched proof of Hadamard Products* (`PoHadProd`)

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $\mathbf{g}_1,\ \mathbf{g}_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
The CRS $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$
A public integer $N \leq M$

Verifier's preprocessed inputs: The elements $\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \mathbf{g}_1^{\mathbf{s}^N}\ \in\ \mathbb{G}_1,\ \ \mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}\ \in\ \mathbb{G}_2$

Common Inputs: Elements $\mathbf{a}_{L,j},\ \mathbf{a}_{R,j},\ \mathbf{a}_{O,j}\ \in\ \mathbb{G}_1 \quad (j = 1, \cdots, k)$

Claim: The Prover knows polynomials $L_j(X),\ R_j(X)$ such that:

$$\mathbf{g}_1^{L_j(\mathbf{s})} = \mathbf{a}_{L,j}\ \ ,\ \ \mathbf{g}_1^{R_j(\mathbf{s})} = \mathbf{a}_{R,j}\ \ ,\ \ \mathbf{g}_1^{L_j \odot R_j(\mathbf{s})} = \mathbf{a}_{O,j}$$

($L_j \odot R_j$ denotes the Hadamard product)

<u>Proof generation</u>

1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates challenges $\gamma,\ \lambda$.
2. $\mathcal{P}$ sends the $\mathbb{F}_p$-element
$$\gamma_\lambda := \sum_{j=1}^{k} \lambda^{j-1} \cdot L_j \odot R_j(\gamma).$$

Randomized sum of twisted products

3. $\mathcal{P}$ computes the polynomial
$$f_{\gamma,\lambda}(X)\ :=\ \Big[\sum_{j=1}^{k} L_j(\gamma \cdot X) \cdot X^N \cdot R_j(X^{-1})\Big]\ -\ \gamma_\lambda \cdot X^N$$

The low degree part

4. $\mathcal{P}$ computes the residue
$$f_{\gamma,\lambda,-}(X) :=\ f_{\gamma,\lambda}(X) \pmod{X^N}$$
and sends the $\mathbb{G}_1$-element $\mathbf{a}_-\ :=\ \mathbf{g}_1^{f_{\gamma,\lambda,-}(\mathbf{s})}$.

Degree upper bound on the low degree part

5. $\mathcal{P}$ computes the polynomial
$$\widehat{f}_{\gamma,\lambda,-}(X) :=\ X^{N-1} \cdot f_{\gamma,\lambda,-}(X^{-1})$$
and sends the $\mathbb{G}_1$-element $\widehat{\mathbf{a}}_-\ :=\ \mathbf{g}_1^{\widehat{f}_{\gamma,\lambda,-}(\mathbf{s})}$.

The high degree part

6. $\mathcal{P}$ computes the polynomial

$$f_{\gamma,\lambda,+}(X) := \sum_{i=N+1}^{\deg(f_{\gamma,\lambda})} \texttt{Coef}(f_{\gamma,\lambda}\ ,\ i) \cdot X^{i-N-1}$$

and sends the $\mathbb{G}_1$-element $\mathbf{a}_+ := \mathbf{g}_1^{f_{\gamma,\lambda,+}(\mathbf{s})}$.

The evaluation challenge

7. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge α.

8. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements

$$\beta_{\gamma,j} := L_j(\gamma \cdot \alpha)\ ,\ \widehat{\beta}_j := R_j(\alpha^{-1}). \quad (j = 1, \cdots, k)$$

9. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements

$$\beta_{\gamma,\lambda,-} := f_{\gamma,\lambda,-}(\alpha)\ ,\ \beta_{\gamma,\lambda,+} := f_{\gamma,\lambda,+}(\alpha).$$

10. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge δ.

The batched divisibility subprotocol

11. $\mathcal{P}$ computes the following polynomials:

(i). $h_1(X) := \sum_{j=1}^{k} \delta^{j-1} \cdot \big[L_j(X) - \beta_{\gamma,j}\big]\ ,\ e_1(X) := X - \gamma^{-1} \cdot \alpha.$

(ii). $h_2(X) := \sum_{j=1}^{k} \delta^{j-1} \cdot \big[R_j(X) - \widehat{\beta}_j\big] + \delta^k \cdot \big[\widehat{f}_{\gamma,\lambda,-}(X) - \alpha^{1-N} \cdot \beta_{\gamma,\lambda,-}\big]\ ,\ e_2(X) := X - \alpha^{-1}.$

(iii). $h_3(X) := \big[f_{\gamma,\lambda,-}(X) - \beta_{\gamma,\lambda,-}\big] + \delta \cdot \big[f_{\gamma,\lambda,+}(X) - \beta_{\gamma,\lambda,+}\big]\ ,\ e_3(X) := X - \alpha$

(iv). $h_4(X) := \big[\sum_{j=1}^{k} \lambda^{j-1} \cdot L_j \odot R_j(X)\big] - \gamma_\lambda\ ,\ e_4(X) := X - \gamma$

12. $\mathcal{P}$ computes the $\mathbb{G}_1$-elements $\mathbf{b}_i := \mathbf{g}_1^{h_i(\mathbf{s})}$ $(i = 1, \cdots, 4)$ as follows:

(i) $\mathbf{b}_1 := \prod_{j=1}^{k} (\mathbf{a}_{L,j})^{\delta^{j-1}} \cdot \mathbf{g}_1^{-\sum_{j=1}^{k} \delta^{j-1} \cdot \beta_{\gamma,j}}$

(ii) $\mathbf{b}_2 := \prod_{j=1}^{k} (\mathbf{a}_{R,j})^{\delta^{j-1}} \cdot (\widehat{\mathbf{a}}_{\gamma,\lambda,-})^{\delta^k} \cdot \mathbf{g}_1^{-\big[[\sum_{j=1}^{k} \delta^{j-1} \cdot \widehat{\beta}_j] + \delta^k \cdot \alpha^{1-N} \cdot \beta_{\gamma,\lambda,-}\big]}$

(iii) $\mathbf{b}_3 := \big[\mathbf{a}_{\gamma,\lambda,-} \cdot \mathbf{a}_{\gamma,\lambda,+}^{\delta}\big] \cdot \mathbf{g}_1^{-[\beta_{\gamma,\lambda,-} + \delta \cdot \beta_{\gamma,\lambda,+}]}$

(iv) $\mathbf{b}_4 := \big[\prod_{j=1}^{k} \mathbf{a}_{O,j}^{\lambda^{j-1}}\big] \cdot \mathbf{g}_1^{-\gamma_\lambda}$.

13. $\mathcal{P}$ sends a proof for the protocol `BatchDiv` on the tuple $[\mathbf{g}_1,\ (\mathbf{b}_i)_{i=1}^4,\ (e_i(X))_{i=1}^4]$

The verification

14. The Verifier $\mathcal{V}$ computes the $\mathbb{G}_1$ elements $\mathbf{b}_i\ \ (i = 1, \cdots, 4)$ as in Step 12.

15. $\mathcal{V}$ verifies the `BatchDiv` subprotocol.

16. $\mathcal{V}$ verifies the equation

$$\sum_{j=1}^{k} \lambda^{j-1} \cdot (\beta_{\gamma,j} \cdot \alpha^N \cdot \widehat{\beta}_j) \ \overset{?}{=}\ \beta_{\gamma,\lambda,-} + \alpha^N \cdot \gamma_\lambda + \alpha^{N+1} \cdot \beta_{\gamma,\lambda,+} \quad \in \quad \mathbb{F}_p.$$

$\square$

The proof consists of 5 $\mathbb{G}_1$ elements and $2k + 5$ $\mathbb{F}_p$ elements. The Prover work is dominated by the 5 $\mathbb{G}_1$ MSMs and the sum of k polynomial products (Step 3) via multimodular FFTs.

Proposition 2.7. *The protocol* `PoHadProd` *is secure in the algebraic group model.*

Proof. Appendix of [Th23]. $\square$

3 Generalizing the permutation argument

For a permutation $\boldsymbol{\sigma} : \ [0,\ N-1] \longrightarrow [0,\ N-1]$, we commit to $\boldsymbol{\sigma}$ by committing to the polynomial

$$S_{\boldsymbol{\sigma}}(X) := \sum_{i=0}^{N-1} \boldsymbol{\sigma}(i) \cdot X^i.$$

In particular, we commit to the identity permutation of $[0,\ N-1]$ by committing to the polynomial $P_{\mathtt{id},N}(X) := \sum_{i=0}^{N-1} k \cdot X^k$.

Similarly, for a (possibly non-injective) map $\boldsymbol{\rho} : \ \mathcal{I} \longrightarrow \mathcal{J}$ of index sets $\subseteq\ [0,\ N-1]$, we commit to $\boldsymbol{\rho}$ by committing to the polynomials

$$S_{\boldsymbol{\rho}}(X) := \sum_{i=0}^{N-1} \boldsymbol{\rho}(i) \cdot X^i \ \ , \ \ \chi_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} X^i.$$

We say polynomials $f(X),\ h(X) \ \in\ \mathbb{F}_p[X]$ are linked by $\boldsymbol{\rho}$ if the coefficients satisfy the equations:

$$\texttt{Coef}(h\ ,\ \boldsymbol{\rho}(i)) \ = \ \texttt{Coef}(f\ ,\ i) \ \ \forall\, i \ \in\ \mathcal{I}. \tag{3.1}$$

We describe a protocol that allows a Prover to succinctly show that two committed polynomials bear this relation. We assume the Verifier stores KZG commitments to the index sets $\mathcal{I}$, $\mathcal{J}$ (i.e. commitments to their indicator polynomials) and to the polynomials

$$S_{\boldsymbol{\rho}}(X) = \sum_{i \in \mathcal{I}} \ \boldsymbol{\rho}(i) \cdot X^i \ \ , \ \ M_{\boldsymbol{\rho}}(X) = \sum_{j \in \mathcal{J}} \ \mathrm{mul}(j, \boldsymbol{\rho}) \cdot X^j \ \ ,$$

where $\mathrm{mul}(j, \boldsymbol{\rho})$ is the cardinality of the pre-image $\{i \in \mathcal{I} : \ \boldsymbol{\rho}(i) = j\}$.

In response to two randomly generated challenges $\delta,\ \alpha\ \in\ \mathbb{F}_p$, the Prover shows that the equation

$$\sum_{i\in\mathcal{I}} \big[\alpha + \texttt{Coef}(f\ ,\ i) + \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}\ =\ \sum_{j\in\mathcal{J}} \mathrm{mul}(j,\boldsymbol{\rho})\cdot\big[\alpha + \texttt{Coef}(h\ ,\ j) + \delta\cdot j\big]^{-1} \tag{3.2}$$

holds. By lemma 1.2, this implies equation 3.1. We now describe the process whereby the Prover shows that equation 3.2 holds.

The Prover verifiably sends commitments to the polynomials

$$f_{\mathcal{I},\alpha,\mathrm{inv}}(X)\ :=\ \sum_{i\in\mathcal{I}} [\texttt{Coef}(f\ ,\ i)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)]^{-1}\cdot X^i$$

$$h_{\mathcal{J},\alpha,\mathrm{inv}}(X)\ :=\ \sum_{j\in\mathcal{J}} \big[\texttt{Coef}(h\ ,\ j)\ +\ \alpha + \sum_{j\in\mathcal{J}} j\cdot X^j\big]^{-1}\cdot X^j.$$

It is straightforward to do so, since they are the unique polynomials of degree $\leq N-1$ that satisfy the Hadamard product equations

$$f_{\mathcal{I},\alpha,\mathrm{inv}}(X)\ \odot\ \big[f(X) + \alpha\cdot[\sum_{k=0}^{N-1} X^i] + \delta\cdot S_{\boldsymbol{\rho}}(X)\big]\ =\ \chi_{\mathcal{I}}(X) \tag{3.3}$$

$$h_{\mathcal{J},\alpha,\mathrm{inv}}(X)\ \odot\ \big[h(X) + \alpha\cdot[\sum_{k=0}^{N-1} X^k] + \delta\cdot\sum_{j\in\mathcal{J}} j\cdot X^j\big]\ =\ \chi_{\mathcal{J}}(X). \tag{3.4}$$

By construction, the left hand side of equation 3.2 is the evaluation $f_{\mathcal{I},\alpha,\mathrm{inv}}(1)$. The right hand side of equation 3.2 is the dot product $h_{\mathcal{J},\alpha,\mathrm{inv}}(X)\circ M_{\boldsymbol{\rho}}(X)$. To that end, the Prover uses the dot product protocol to show that these $\mathbb{F}_p$-elements coincide, i.e.

$$f_{\mathcal{I},\alpha,\mathrm{inv}}(1)\ =\ h_{\mathcal{J},\alpha,\mathrm{inv}}(X)\ \circ\ M_{\boldsymbol{\rho}}(X)\ =\ h_{\mathcal{J},\alpha,\mathrm{inv}}\odot M_{\boldsymbol{\rho}}(1).$$

This is an argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{SelfMap}}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{C}_{\boldsymbol{\rho}},\ \mathbf{C}_{\mathcal{I}}),\ \mathbf{b}] = \left\{\begin{array}{l} (\mathbf{a},\mathbf{b}\in\mathbb{G}_1),\ f(X),\ h(X)\ \in\ \mathbb{F}_p[X]) : \\ \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}\ ,\ \mathbf{g}_1^{h(\mathbf{s})} = \mathbf{b} \\ \texttt{Coef}\big(h\ ,\ \boldsymbol{\rho}(i)\big) = \texttt{Coef}\big(f\ ,\ i\big)\ \forall\ i\in\mathcal{I} \end{array}\right\}$$

Here, $\boldsymbol{\rho}$ is a map $\mathcal{I}\longrightarrow\mathcal{J}$ of index sets $\mathcal{I},\ \mathcal{J}\subseteq\ [0,\ N-1]$. $\mathbf{C}_{\boldsymbol{\rho}},\ \mathbf{C}_{\mathcal{I}}$ are the KZG commitment to the polynomials $S_{\boldsymbol{\rho}}(X) := \sum_{i\in\mathcal{I}}\boldsymbol{\rho}(i)\cdot X^i,\ \chi_{\mathcal{I}}(X) := \sum_{i\in\mathcal{I}} X^i$ respectively.

Protocol 3.1. *Proof of self-map* (`PoSelfMap`)

Parameters: A pairing $\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\longrightarrow\mathbb{G}_T$; generators $\mathbf{g}_1,\ \mathbf{g}_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively

The CRS $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots\ ,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

A public integer $N\leq M$

Common preprocessed input: The polynomials

$$P_{\mathrm{id}}(X) := \sum_{i=0}^{N-1} i\cdot X^i\ \ ,\ \ S_{\boldsymbol{\rho}}(X) := \sum_{i=0}^{N-1}\boldsymbol{\rho}(i)\cdot X^i\ \ ,\ \ M_{\boldsymbol{\rho}}(X) := \sum_{j\in\mathcal{J}}\mathrm{mul}(j,\boldsymbol{\rho})\cdot X^j$$

for index sets $\mathcal{I},\ \mathcal{J}\ \subseteq\ [0,\ N-1]$ and a map $\boldsymbol{\rho} : \mathcal{I} \longrightarrow \mathcal{J}$ with domain $\mathcal{I}$ and image $\mathcal{J}$.

Verifier's preprocessed input: The elements $\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \mathbf{g}_1^{\mathbf{s}^N}\ \in\ \mathbb{G}_1,\quad \mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}\ \in\ \mathbb{G}_2$
The [KZG10] commitments

$$\mathbf{g}_1^{\mathbf{s}^N}\ ,\ \mathbf{C}_{\boldsymbol{\rho}} := \mathbf{g}_1^{S_{\boldsymbol{\rho}}(\mathbf{s})}\ ,\ \mathbf{C}_1 := \mathbf{g}_1^{\sum_{i-0}^{N-1}\mathbf{s}^i}\ ,\ \mathbf{M}_{\boldsymbol{\rho}} := \mathbf{g}_1^{M_{\boldsymbol{\rho}}(\mathbf{s})}$$

$$\mathbf{C}_{\text{id}} := \mathbf{g}_1^{P_{\text{id}}(\mathbf{s})}\ ,\ \mathbf{C}_{\text{id},\mathcal{J}} := \mathbf{g}_1^{P_{\text{id}}\odot\chi_{\mathcal{J}}(\mathbf{s})} = \mathbf{g}_1^{\sum\limits_{j\in\mathcal{J}} j\cdot\mathbf{s}^j}\ .$$

Common Inputs: Elements $\mathbf{a},\ \mathbf{b}\ \in\ \mathbb{G}_1$

Claim: The Prover knows polynomials $f(X)$, $h(X)$ of degree $\leq N-1$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ ,\ \mathbf{b} = \mathbf{g}_1^{h(\mathbf{s})}\ ,\ \texttt{Coef}(h\ ,\ \boldsymbol{\rho}(i))\ =\ \texttt{Coef}(f\ ,\ i)\ \ \forall\, i\in\ \mathcal{I}$$

Proof generation

1. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates challenges δ, α.
2. $\mathcal{P}$ computes the polynomials
$$f_{\mathcal{I},\alpha,\text{inv}}(X)\ :=\ \sum_{i\in\mathcal{I}}\ \big[\texttt{Coef}(f\ ,\ i)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}\cdot X^i$$
$$h_{\mathcal{J},\delta,\alpha,\text{inv}}(X)\ :=\ \sum_{j\in\mathcal{J}}\ \big[\texttt{Coef}(h\ ,\ j)\ +\ \alpha+\delta\cdot j\big]^{-1}\cdot X^j.$$
3. $\mathcal{P}$ sends the $\mathbb{G}_1$-elements
$$\mathbf{a}_{\mathcal{I},\delta,\alpha,\text{inv}}\ :=\ \mathbf{g}_1^{f_{\mathcal{I},\delta,\alpha,\text{inv}}(\mathbf{s})}\ ,\ \mathbf{b}_{\mathcal{J},\delta,\alpha,\text{inv}}\ := \mathbf{g}_1^{h_{\mathcal{J},\delta,\alpha,\text{inv}}(\mathbf{s})}.$$
4. $\mathcal{P}$ sends the element
$$\beta_{\mathcal{I},\delta,\alpha,\text{inv}} := f_{\mathcal{I},\delta,\alpha,\text{inv}}(1)\ \in\ \mathbb{F}_p.$$
5. $\mathcal{P}$ sends (batched) proofs of the following Hadamard and dot product protocols:

- $\texttt{PoHadProd}\big[\mathbf{g}_1,\ \big(\mathbf{a}_{\mathcal{I},\delta,\alpha,\text{inv}},\ \mathbf{a}\cdot\mathbf{C}_1^{\alpha}\cdot\mathbf{C}_{\boldsymbol{\rho}}^{\delta}\big),\ \mathbf{C}_{\mathcal{I}}\big]$
- $\texttt{PoHadProd}\big[\mathbf{g}_1,\ \big(\mathbf{b}_{\mathcal{J},\delta,\alpha,\text{inv}},\ \mathbf{b}\cdot\mathbf{C}_1^{\alpha}\cdot\mathbf{C}_{\text{id},\mathcal{J}}^{\delta}\big),\ \mathbf{C}_{\mathcal{J}}\big]$
- $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{M}_{\boldsymbol{\rho}},\ \mathbf{b}_{\mathcal{J},\delta,\alpha,\text{inv}}),\ \beta_{\mathcal{I},\delta,\alpha,\text{inv}}]$.

6. $\mathcal{P}$ sends (batched) proofs of the following degree upper bound protocols:

- $\texttt{PoDegUp}[\mathbf{g}_1,\ \mathbf{a},\ N-1]$
- $\texttt{PoDegUp}[\mathbf{g}_1,\ \mathbf{b},\ N-1]$
- $\texttt{PoDegUp}[\mathbf{g}_1,\ \mathbf{a}_{\mathcal{I},\delta,\alpha,\text{inv}},\ N-1]$
- $\texttt{PoDegUp}[\mathbf{g}_1,\ \mathbf{b}_{\mathcal{J},\delta,\alpha,\text{inv}},\ N-1]$.

7. $\mathcal{P}$ sends the $\mathbb{G}_1$-element

$$\mathbf{Q}_{\mathcal{I},\delta,\alpha,\mathrm{inv}} \;:=\; \mathbf{g}_1^{[f_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(\mathbf{s})-\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}]\big/[\mathbf{s}-1]} \;\in\; \mathbb{G}_1.$$

The verification

8. The Verifier $\mathcal{V}$ verifies the proofs of the degree upper bounds (`PoDegUp`), the Hadamard product protocols (`PoHadProds`) and the dot product protocol (`PoDotProd`).

9. $\mathcal{V}$ verifies the equation

$$(\mathbf{Q}_{\mathcal{I},\delta,\alpha,\mathrm{inv}})^{\mathbf{s}-1} \;\overset{?}{=}\; \mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}} \cdot \mathbf{g}_1^{-\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}} \;\in\; \mathbb{G}_1$$

via the pairing check $\mathbf{e}(\mathbf{Q}_{\mathcal{I},\delta,\alpha,\mathrm{inv}}\;,\;\mathbf{g}_2^{\mathbf{s}-1}) \;\overset{?}{=}\; \mathbf{e}(\mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}} \cdot \mathbf{g}_1^{-\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}}\;,\;\mathbf{g}_2)$. □

In practice, the Hadamard and dot products should be batched so as to minimize the number of $\mathbb{G}_1$-elements in the proof. Likewise, the degree upper bounds - including those arising from the Hadamard and dot products - should be batched.

4 A protocol for weighted sums

As before, let $\mathcal{I}$, $\mathcal{J}$ be index sets $\subseteq$ $[0,\; N-1]$ and let $\boldsymbol{\rho} : \mathcal{I} \longrightarrow \mathcal{J}$ be a map with domain $\mathcal{I}$ and image $\mathcal{J}$. For every index $j \in \mathcal{J}$, we denote by $\boldsymbol{\rho}^{-1}(j)$ the pre-image:

$$\boldsymbol{\rho}^{-1}(j) \;:=\; \{i \in \mathcal{I} \;:\; \boldsymbol{\rho}(i) = j\}.$$

We denote the cardinality of $\boldsymbol{\rho}^{-1}(j)$ by $\mathrm{mul}(j, \boldsymbol{\rho})$. We commit to the map $\boldsymbol{\rho}$ by committing to the polynomial

$$S_{\boldsymbol{\rho}}(X) \;:=\; \sum_{i\in\mathcal{I}} \boldsymbol{\rho}(i) \cdot X^i$$

and to the indicator polynomial $\chi_{\mathcal{I}}(X)$ of $\mathcal{I}$ via the KZG commitment scheme.

Let $W(X)$ be a committed polynomial of “weights”. For committed polynomials $f(X)$, $h(X)$, we describe a protocol that allows a Prover to succinctly show that the following equations hold:

$$\texttt{Coef}(h\;,\;j) \;=\; \sum_{i\in\boldsymbol{\rho}^{-1}(j)} \texttt{Coef}(W\;,\;i) \cdot \texttt{Coef}(f\;,\;i) \quad \forall\, j \;\in\; \mathcal{J}. \tag{4.1}$$

We assume that the Verifier stores the KZG commitments

$$\mathbf{g}_1^{\mathbf{s}^N} \;\;,\;\; \mathbf{C}_{\boldsymbol{\rho}} := \mathbf{g}_1^{S_{\boldsymbol{\rho}}(\mathbf{s})} \;\;,\;\; \mathbf{C}_1 := \mathbf{g}_1^{\sum_{i=0}^{N-1}\mathbf{s}^i} \;\;,\;\; \mathbf{M}_{\boldsymbol{\rho}} := \mathbf{g}_1^{M_{\boldsymbol{\rho}}(\mathbf{s})} \;\;,\;\; \mathbf{W} := \mathbf{g}_1^{W(\mathbf{s})}$$

$$\mathbf{C}_{\mathrm{id}} := \mathbf{g}_1^{P_{\mathrm{id}}(\mathbf{s})} \;\;,\;\; \mathbf{C}_{\mathrm{id},\mathcal{J}} := \mathbf{g}_1^{P_{\mathrm{id}}\odot\chi_{\mathcal{J}}(\mathbf{s})} = \mathbf{g}_1^{\sum_{j\in\mathcal{J}} j\cdot\mathbf{s}^j}\;.$$

Note that because of the Schwartz-Zippel lemma, equation 4.1 reduces to showing that for a randomly generated challenge $\zeta \;\in\; \mathbb{F}_p$, the following equation holds:

$$\sum_{j\in\mathcal{J}} \texttt{Coef}(h\;,\;j) \cdot \zeta^j \;=\; \sum_{i\in\mathcal{I}} \texttt{Coef}(W\;,\;i) \cdot \texttt{Coef}(f\;,\;i) \cdot \zeta^{\boldsymbol{\rho}(i)}\;. \tag{4.2}$$

The left hand side of Equation 4.2 is merely the dot product $h(X) \circ \chi_{\mathcal{J}}(\zeta \cdot X)$, which can be verifiably sent to the Verifier using the dot product protocol. The right hand side of Equation 4.2 is the polynomial

$$\big[f(X) \odot W(X)\big] \circ \big[\sum_{i \in \mathcal{I}} \zeta^{\boldsymbol{\rho}(i)} \cdot X^i\big] \ ,$$

where $\odot$ denotes the Hadamard product and $\circ$ denotes the dot product. This part is marginally more subtle, but it boils down to verifiably sending a commitment to the polynomial

$$S_{\boldsymbol{\rho},\zeta}(X) \ := \ \sum_{i \in \mathcal{I}} \zeta^{\boldsymbol{\rho}(i)} \cdot X^i,$$

followed by invoking the Hadamard product and dot product protocols. The polynomial $S_{\boldsymbol{\rho},\zeta}(X)$ is the unique polynomial of degree $\leq \ N-1$ with the following properties:

- $S_{\boldsymbol{\rho},\zeta}(X) \odot \chi_{\mathcal{I}}(X) \ = \ S_{\boldsymbol{\rho},\zeta}(X)$
- $\chi_{\mathcal{J}}(\zeta \cdot X)$ is obtained by the action of the map $\boldsymbol{\rho}$ on the polynomial $S_{\boldsymbol{\rho},\zeta}(X)$, i.e.

$$\texttt{Coef}\big(S_{\boldsymbol{\rho},\zeta} \ , \ i\big) \ = \ \texttt{Coef}\big(\chi_{\mathcal{J}}(\zeta \cdot X) \ , \ \boldsymbol{\rho}(i)\big) \quad \forall \ i \ \in \ \mathcal{I}.$$

Thus, the task of succinctly proving equation 4.2 reduces to the the proof of twist (`PoTwist`) and the self-map protocol (`PoSelfMap`) from the preceding section.

Protocol 4.1. *Proof of weighted sums* (`PoWeightedSums`)

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $\mathbf{g}_1, \ \mathbf{g}_2$ for $\mathbb{G}_1, \ \mathbb{G}_2$ respectively
The CRS $[\mathbf{g}_1, \ \mathbf{g}_1^{\mathbf{s}}, \ \cdots \ , \ \mathbf{g}_1^{\mathbf{s}^M}] \ , \ [\mathbf{g}_2, \ \mathbf{g}_2^{\mathbf{s}}]$
A public integer $N \leq M$

Common preprocessed input: A public integer N; the polynomials

$$\sum_{i=0}^{N-1} X^i \ , \ P_{\text{id}}(X) := \sum_{i=0}^{N-1} i \cdot X^i \ , \ S_{\boldsymbol{\rho}}(X) := \sum_{i=0}^{N-1} \boldsymbol{\rho}(i) \cdot X^i \ , \ M_{\boldsymbol{\rho}}(X) := \sum_{j \in \mathcal{J}} \text{mul}(j, \boldsymbol{\rho}) \cdot X^j$$

for index sets $\mathcal{I}, \ \mathcal{J} \ \subseteq \ [0, \ N-1]$ and a map $\boldsymbol{\rho} : \mathcal{I} \longrightarrow \mathcal{J}$ with domain $\mathcal{I}$ and image $\mathcal{J}$.
The polynomial $W(X)$ of weights such that $W(X) \odot \chi_{\mathcal{I}}(X) = W(X)$.

Verifier's preprocessed input: The elements $\mathbf{g}_1, \ \mathbf{g}_1^{\mathbf{s}}, \ \mathbf{g}_1^{\mathbf{s}^N} \ \in \ \mathbb{G}_1, \quad \mathbf{g}_2, \ \mathbf{g}_2^{\mathbf{s}} \ \in \ \mathbb{G}_2$
The KZG commitments

$$\mathbf{g}_1^{\mathbf{s}^N} \ , \ \mathbf{C}_{\boldsymbol{\rho}} := \mathbf{g}_1^{S_{\boldsymbol{\rho}}(\mathbf{s})} \ , \ \mathbf{C}_1 := \mathbf{g}_1^{\sum_{i=0}^{N-1} \mathbf{s}^i} \ , \ \mathbf{M}_{\boldsymbol{\rho}} := \mathbf{g}_1^{M_{\boldsymbol{\rho}}(\mathbf{s})} \ , \ \mathbf{W} := \mathbf{g}_1^{W(\mathbf{s})}$$

$$\mathbf{C}_{\text{id}} := \mathbf{g}_1^{P_{\text{id}}(\mathbf{s})} \ , \ \mathbf{C}_{\text{id},\mathcal{J}} := \mathbf{g}_1^{P_{\text{id}} \odot \chi_{\mathcal{J}}(\mathbf{s})} = \mathbf{g}_1^{\sum_{j \in \mathcal{J}} j \cdot \mathbf{s}^j} \ .$$

Common Inputs: Elements $\mathbf{a}, \ \mathbf{b} \ \in \ \mathbb{G}_1$

Claim: The Prover knows polynomials $f(X), \ h(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})} \ , \ \mathbf{b} = \mathbf{g}_1^{h(\mathbf{s})}$$

$$\texttt{Coef}(h\ ,\ j)\ =\ \sum_{i\in\boldsymbol{\rho}^{-1}(j)} \texttt{Coef}(W\ ,\ i)\cdot\texttt{Coef}(f\ ,\ i) \qquad \forall\, j\ \in\ \mathcal{J}$$

where $\boldsymbol{\rho}^{-1}(j)$ denotes the pre-image set $\{i \in \mathcal{I} :\ \boldsymbol{\rho}(i) = j\}$.

Proof generation

1. The Prover $\mathcal{P}$ sends the $\mathbb{G}_1$-element

$$\mathbf{a}_W\ :=\ \mathbf{g}_1^{f\odot W(\mathbf{s})}.$$

2. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge ζ.
3. $\mathcal{P}$ computes the polynomial $\chi_{\mathcal{J}}(\zeta\cdot X)\ :=\ \sum_{j\in\mathcal{J}} \zeta^j \cdot X^j$ and sends the $\mathbb{G}_1$-element

$$\mathbf{C}_{\mathcal{J},\zeta}\ :=\ \mathbf{g}_1^{\chi_{\mathcal{J}}(\zeta\cdot\mathbf{s})}\ =\ \mathbf{g}_1^{\sum\limits_{j\in\mathcal{J}} \zeta^j\cdot\mathbf{s}^j}.$$

4. $\mathcal{P}$ computes the polynomial

$$S_{\boldsymbol{\rho},\zeta}(X)\ :=\ \sum_{i\in\mathcal{I}} \zeta^{\boldsymbol{\rho}(i)}\cdot X^i$$

and sends the $\mathbb{G}_1$-element

$$\mathbf{C}_{\boldsymbol{\rho},\zeta}\ :=\ \mathbf{g}_1^{S_{\boldsymbol{\rho},\zeta}(\mathbf{s})}.$$

5. $\mathcal{P}$ sends the $\mathbb{F}_p$-element

$$\beta\ :=\ \sum_{j\in\mathcal{J}} \texttt{Coef}(h\ ,\ j)\cdot\zeta^j\ =\ h\odot\chi_{\mathcal{J}}(\zeta)$$

6. $\mathcal{P}$ sends a proof of $\texttt{PoTwist}[\mathbf{g}_1,\ (\mathbf{C}_{\mathcal{J}},\ \zeta),\ \mathbf{C}_{\mathcal{J},\zeta}]$. (proof of twist)
7. $\mathcal{P}$ sends (batched) proofs of the following Hadamard and dot products:

- $\texttt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{W}),\ \mathbf{a}_W]$.
- $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{b},\ \mathbf{C}_{\mathcal{J},\zeta}),\ \beta]$
- $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{a}_W,\ \mathbf{C}_{\boldsymbol{\rho},\zeta}),\ \beta]$.

8. $\mathcal{P}$ sends a proof of $\texttt{PoSelfMap}[\mathbf{g}_1,\ (\mathbf{C}_{\boldsymbol{\rho},\zeta},\ \mathbf{C}_{\boldsymbol{\rho}},\ \mathbf{C}_{\mathcal{I}}),\ \mathbf{C}_{\mathcal{J},\zeta}]$ (self-map protocol).
9. $\mathcal{P}$ sends a proof of $\texttt{PoDegUp}[\mathbf{g}_1,\ (\mathbf{C}_{\boldsymbol{\rho},\zeta},\ N-1)]$. (degree upper bound protocol).

The verification

10. The Verifier $\mathcal{V}$ verifies the subprotocols $\texttt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{W}),\ \mathbf{a}_W]$,

$\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{b},\ \mathbf{C}_{\mathcal{J},\zeta}),\ \beta]$, $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{a}_W,\ \mathbf{C}_{\boldsymbol{\rho},\zeta}),\ \beta]$,

$\texttt{PoSelfMap}[\mathbf{g}_1,\ (\mathbf{C}_{\boldsymbol{\rho},\zeta},\ \mathbf{C}_{\boldsymbol{\rho}}),\ \mathbf{C}_{\mathcal{J},\zeta}]$, $\texttt{PoDegUp}[\mathbf{g}_1,\ (\mathbf{C}_{\boldsymbol{\rho},\zeta},\ N-1)]$. □

In practice, the Hadamard and dot products - including those arising from the self-map subprotocol - should be batched so as to have fewer $\mathbb{G}_1$-elements in the proof (and hence, fewer

MSMs in the proof generation). The proof of twist boils down to a polynomial commitment opening and can be batched with the other openings using the protocol `BatchDiv`. The protocol `PoWeightedSums` can be merged with the Snark described in [Th23], in which case the only $\mathbb{G}_1$-elements needed in the proof are the [KZG10] commitments to the polynomials $f \odot W(X)$, $\chi_{\mathcal{J}}(\zeta \cdot X)$, and

$$\sum_{i \in \mathcal{I}} \left[\zeta^{\boldsymbol{\rho}(i)} + \delta \cdot \boldsymbol{\rho}(i) + \alpha\right]^{-1} \cdot X^i \ , \ \sum_{j \in \mathcal{J}} \left[\zeta^{j} + \delta \cdot \boldsymbol{\rho}(j) + \alpha\right]^{-1} \cdot X^j \ ,$$

where $\delta,\ \alpha \in \mathbb{F}_p$ are challenges generated after the commitment to $S_{\boldsymbol{\rho},\zeta}(X)$ has been sent (as in the preceding protocol `PoSelfMap`).

References

[BCKL21] E. Ben-Sasson, D. Carmon, S. Kopparty, D. Levit, *Elliptic Curve Fast Fourier Transform (ECFFT) Part I: Fast Polynomial Algorithms over all Finite Fields*, https://arxiv.org/abs/2107.08473

[Bl22] R. Bloemen, *NTT transform argument*, https://xn–2-umb.com/22/ntt-argument/

[CHMMVW20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely and N.P. Ward. *Marlin: Preprocessing zk-SNARKs with universal and updatable SRS.* Eurocrypt 2020, Part I, volume 12105 of LNCS

[EFG22] L. Eagen, D. Fiore, and A. Gabizon. *cq: Cached quotients for fast lookups*, https://eprint.iacr.org/2022/1763

[EF23] L. Eagen and A. Gabizon, *cqLin: Efficient linear operations on KZG commitments with cached quotients*, https://eprint.iacr.org/2023/393

[FST06] D. Freeman, M. Scott, E. Teske, *A taxonomy of pairing-friendly elliptic curves*

[FS87] A. Fiat, A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems.* In Andrew M. Odlyzko, editor, Crypto'86, volume 263 of LNCS

[FK] D. Feist and D. Khovratovich. *Fast amortized Kate proofs*, https://eprint.iacr.org/2023/033

[FKL18] G. Fuchsbauer, E. Kiltz,and J. Loss. *The algebraic group model and its applications.* In Advances in Cryptology - Crypto 2018- 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 1923, 2018, Proceedings, Part II, pages 33–62, 2018.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. *Quadratic span programs and succinct NIZKs without PCPs.* In Advances in Cryptology - Eurocrypt 2013

[GWC19] A. Gabizon, Z. Williamson, O. Ciobotoru, *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*, https://eprint.iacr.org/2019/953

[GW20] A. Gabizon, Z. Williamson, *Plookup: A simplified polynomial protocol for lookup tables*, https://eprint.iacr.org/2020/315.pdf

[Hab22] U. Habock, *Multivariate lookups based on logarithmic derivatives*, https://eprint.iacr.org/2022/1530

[KS98] E. Kaltofen and V. Shoup. *Subquadratic-time factoring of polynomials over finite fields.* Mathematics of computation, 67(223):1179–1197, 1998

[KZG10] A. Kate, G. Zaverucha, and I. Goldberg. *Constant-size commitments to polynomials and their applications.* In Masayuki Abe, editor, Asiacrypt 2010, volume 6477 of LNCS, pages 177–194. Springer, Heidelberg, December 2010.

[Mil86] V. Miller, *Short Programs for functions on Curves*

[Sh01] V. Shoup, *The NTL Library*, https://libntl.org/

[SS71] A. Schönhage, V. Strassen. *Schnelle multiplikation großer zahlen.* Computing, 7(34):281–292,197.

[Th23] S. Thakur, *A flexible Snark via the monomial basis*, https://eprint.iacr.org/2023/1255

A Deferred proofs

A.1 Self-map protocol

Proposition A.1. *The protocol* `PoSelfMap` *is secure in the algebraic group model.*

Proof. (Sketch) Since completeness is straightforward, it suffices to demonstrate soundness. Suppose a PPT algorithm $\mathcal{A}_{\texttt{PPT}}$ outputs an accepting transcript.

Thus, in particular, the following subprotocols are validated by the Verifier:

- $\texttt{PoHadProd}\big[\mathbf{g}_1,\ \big(\mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}},\ \mathbf{a}\cdot\mathbf{C}_1^{\alpha}\cdot\mathbf{C}_{\boldsymbol{\rho}}^{\delta}\big),\ \mathbf{C}_{\mathcal{I}}\big]$
- $\texttt{PoHadProd}\big[\mathbf{g}_1,\ \big(\mathbf{b}_{\mathcal{J},\delta,\alpha,\mathrm{inv}},\ \mathbf{b}\cdot\mathbf{C}_1^{\alpha}\cdot\mathbf{C}_{\mathrm{id},\mathcal{J}}^{\delta}\big),\ \mathbf{C}_{\mathcal{J}}\big]$

These Hadamard product subprotocols - in conjunction with the degree upper bound protocols - imply that with overwhelming probability, an extractor $\mathcal{E}_{\texttt{PPT}}$ can simulate the extractors of the Hadamard and dot product protocols to extract polynomials $f^*(X),\ h^*(X),\ f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(X)$, $h^*_{\mathcal{J},\delta,\alpha,\mathrm{inv}}(X)$ of degree $\leq N-1$ such that:

$$\mathbf{a}=\mathbf{g}_1^{f^*(\mathbf{s})}\ \ ,\ \ \mathbf{b}=\mathbf{g}_1^{h^*(\mathbf{s})}\ \ ,\ \ \mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}}=\mathbf{g}_1^{f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(\mathbf{s})}\ \ ,\ \ \mathbf{b}_{\mathcal{J},\delta,\alpha,\mathrm{inv}}=\mathbf{g}_1^{h^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(\mathbf{s})}$$

and such that these extracted polynomials satisfy the following equations:

1. $f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(X)\ \odot\ \big[f(X)+\delta\cdot[\sum\limits_{i\in\mathcal{I}}\boldsymbol{\rho}(i)\cdot X^i]+\alpha\cdot\sum_{k=0}^{N-1}X^k\big]\ =\ \chi_{\mathcal{I}}(X)$ or equivalently,

$$f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(X)\ =\ \sum_{i\in\mathcal{I}}\big[\texttt{Coef}(f^*\ ,\ i)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}\cdot X^i.$$

2. $h^*_{\mathcal{J},\delta,\alpha,\mathrm{inv}}(X)\ \odot\ \big[[h^*(X)+\sum\limits_{j\in\mathcal{J}}j\cdot X^j]+\alpha\cdot\sum_{k=0}^{N-1}X^k\big]\ =\ \chi_{\mathcal{J}}(X)$ or equivalently,

$$h^*_{\mathcal{J},\delta,\alpha,\mathrm{inv}}(X)\ =\ \sum_{j\in\mathcal{J}}\big[\texttt{Coef}(h^*\ ,\ j)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}\cdot X^j.$$

Furthermore, the dot product subprotocol $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{M}_{\boldsymbol{\rho}},\ \mathbf{b}_{\mathcal{J},\delta,\alpha,\mathrm{inv}}),\ \beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}]$ implies that with overwhelming probability, the extracted polynomials $f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(X),\ \ h^*_{\mathcal{J},\delta,\alpha,\mathrm{inv}}(X)$ satisfy the equation

$$\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}\ =\ M_{\boldsymbol{\rho}}(X)\ \circ\ h^*_{\mathcal{J},\delta,\alpha,\mathrm{inv}}(X)\ =\ \sum_{j\in\mathcal{J}}\mathrm{mul}(j,\boldsymbol{\rho})\cdot\big[\texttt{Coef}(f^{\boldsymbol{\rho}}\ ,\ j)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}.$$

The pairing check

$$\mathbf{e}(\mathbf{Q}_{\mathcal{I},\delta,\alpha,\mathrm{inv}}\ ,\ \mathbf{g}_2^{\mathbf{s}-1})\ \stackrel{?}{=}\ \mathbf{e}(\mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}}\cdot\mathbf{g}_1^{-\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}}\ ,\ \mathbf{g}_2)$$

implies that with overwhelming probability, the extracted polynomial $f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(X)$ satisfies the equation

$$\beta_{\mathcal{I},\delta,\alpha,\mathrm{inv}}=f^*_{\mathcal{I},\delta,\alpha,\mathrm{inv}}(1).$$

Thus, with overwhelming probability, the coefficients of the extracted polynomials $f^*(X)$, $h^*(X)$ satisfy the equation

$$\sum_{j\in\mathcal{J}}\mathrm{mul}(j,\boldsymbol{\rho})\cdot\big[\texttt{Coef}(h^*\ ,\ j)\ +\ \alpha\ +\ \delta\cdot j\big]^{-1}\ =\ \sum_{i\in\mathcal{I}}\big[\texttt{Coef}(f^*\ ,\ i)\ +\ \alpha\ +\ \delta\cdot\boldsymbol{\rho}(i)\big]^{-1}.$$

Since the challenges $\alpha,\ \delta$ were randomly and uniformly generated after the elements $\mathbf{a}_{\mathcal{I},\delta,\alpha,\mathrm{inv}}$, $\mathbf{b}_{\mathcal{J},\delta,\alpha,\mathrm{inv}}$ were sent, lemma 1.2 implies that with op, the extracted polynomials $f^*(X),\ h^*(X)$ bear the relation

$$\texttt{Coef}(f^*\ ,\ i)\ =\ \texttt{Coef}(h^*\ ,\ \boldsymbol{\rho}(i))\ \ \forall\, i \in \mathcal{I},$$

which completes the proof of soundness. □

A.2 Protocol for weighted sums

Proposition A.2. *The protocol* `PoWeightedSums` *is secure in the algebraic group model.*

Proof. (Sketch) Since completeness is straightforward, it suffices to demonstrate soundness. Suppose a PPT algorithm $\mathcal{A}_{\texttt{PPT}}$ outputs an accepting transcript.

The subprotocol $\texttt{PoTwist}[\mathbf{g}_1,\ (\mathbf{C}_{\mathcal{J}},\ \zeta),\ \mathbf{C}_{\mathcal{J},\zeta}]$ (proof of twist) implies that with overwhelming probability,

$$\mathbf{C}_{\mathcal{J},\zeta}\ =\ \mathbf{g}_1^{\chi_{\mathcal{I}}(\zeta\cdot\mathbf{s})}\ =\ \mathbf{g}_1^{\sum\limits_{j\in\mathcal{J}}\zeta^j\cdot\mathbf{s}^j}\ .$$

Furthermore, the subprotocol $\texttt{PoSelfMap}[\mathbf{g}_1,\ (\mathbf{C}_{\boldsymbol{\rho},\zeta},\ \mathbf{C}_{\boldsymbol{\rho}}),\ \mathbf{C}_{\mathcal{J},\zeta}]$ (self-map protocol) implies that with overwhelming probability, an extractor $\mathcal{E}_{\texttt{PPT}}$ can simulate the extractor of `PoSelfMap` to extract a polynomial $S^*_{\boldsymbol{\rho},\zeta}(X)$ such that

$$\mathbf{C}_{\boldsymbol{\rho},\zeta}\ =\ \mathbf{g}_1^{S^*_{\boldsymbol{\rho},\zeta}(\mathbf{s})}\ \ ,\ \ \texttt{Coef}(S^*_{\boldsymbol{\rho},\zeta}\ ,\ i) = \zeta^{\boldsymbol{\rho}(i)}\quad \forall\ i\ \in\ \mathcal{I}.$$

The Hadamard and dot product subprotocols

- $\texttt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{W}),\ \mathbf{a}_W]$,
- $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{a}_W,\ \mathbf{C}_{\boldsymbol{\rho},\zeta}),\ \beta]$
- $\texttt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{b},\ \mathbf{C}_{\mathcal{J},\zeta}),\ \beta]$

imply that with overwhelming probability, $\mathcal{E}_{\texttt{PPT}}$ can simulate the extractors of the Hadamard and dot product protocols to extract polynomials $f^*(X),\ h^*(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f^*(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{h^*(\mathbf{s})}\ \ ,\ \ \mathbf{a}_W = \mathbf{g}_1^{f^*\odot W(\mathbf{s})}$$

and

$$\text{(A.1)}\qquad f^*(X)\circ\chi_{\mathcal{J}}(\zeta\cdot X)\ =\ \beta\ =\ [f^*(X)\odot W(X)]\circ S^*_{\boldsymbol{\rho},\zeta}(X).$$

The left hand side of equation A.1 is $\sum_{j\in\mathcal{J}}\texttt{Coef}(h^*\ ,\ j)\cdot\zeta^j$, while the right hand side is $\sum_{i\in\mathcal{I}}\texttt{Coef}(f^*\ ,\ i)\cdot\texttt{Coef}(W\ ,\ i)\cdot\zeta^{\boldsymbol{\rho}(i)}$. Since the challenge ζ was randomly and uniformly generated, the Schwartz-Zippel lemma implies that with overwhelming probability, the coefficients of the extracted polynomials satisfy the equations

$$\texttt{Coef}(h^*\ ,\ j)\ =\ \sum_{i\in\boldsymbol{\rho}^{-1}(j)}\texttt{Coef}(W\ ,\ i)\cdot\texttt{Coef}(f^*\ ,\ i)\qquad\forall\ j\ \in\ \mathcal{J}\ ,$$

which completes the proof of soundness. □

B Multimodular FFTs

A consequence of using the monomial basis is that the only superlinear computations the Prover performs are products of polynomial in $\mathbb{F}_p[X]$. The simplest and the most efficient way to do so is to use multimodular FFTs (terminology as in the NTL library).

We fix highly 2-adic primes $p_1,\ p_2$ such that $p < \min(p_1, p_2)$ and the product $p_1 \cdot p_2$ is larger than $p^2 \cdot M$ where M is an upper bound on the degrees of any polynomials to be multiplied. Given polynomials $f_1(X),\ f_2(X),\ \in\ \mathbb{F}_p[X]$, a the Prover computes the product $f_1(X) \cdot f_2(X)$ as follows.

1. Lift the polynomial $f_1(X),\ f_2(X)$ to characteristic zero. Denote these polynomials by $\widetilde{f_1}(X)$, $\widetilde{f_2}(X)$.

2. Compute the polynomials $\widetilde{f_1}(X) \cdot \widetilde{f_2}(X) \pmod{p_i}$ using FFTs in $\mathbb{F}_{p_i}$ for $i = 1,\ 2$.

3. Use the Chinese remainder theorem on these polynomials to recover the polynomial $\widetilde{f_{1,2}}(X) := \widetilde{f_1}(X) \cdot \widetilde{f_2}(X) \in \mathbb{Z}[X]$.

4. Reduce the coefficients of $\widetilde{f_{1,2}}(X)$ modulo p to retrieve the $\mathbb{F}_p$-polynomial $f_1(X) \cdot f_2(X)$.

Computing a sum $\sum_{j=1}^{k} f_{j,1}(X) \cdot f_{j,2}(X)$ entails k FFTs and a single inverse FFT *per prime modulus used* in addition to the Chinese remainder theorem and reduction of the coefficients modulo p.

Steve Thakur
Panther Protocol Cryptography Team
Email: steve@pantherprotocol.io,
stevethakur01@gmail.com