DART: Distributed Argument of knowledge for Rough Terrains

Steve Thakur

Mozak

Abstract

We describe a fully distributed KZG-based Snark instantiable with any pairing-friendly curve with a sufficiently large scalar field. In particular, the proof system is compatible with Cocks-Pinch or Brezing-Weng outer curves to the the widely used curves such as secp256k1, ED25519, BLS12-381 and BN254.

This allows us to retain the fully parallelizable nature and the $O(1)$ communication complexity of *Pianist* ([LXZ+23]) in conjunction with circumventing the huge overhead of non-native arithmetic for prominent use cases such as scalar multiplications and/or pairings for Bitcoin (secp256k1), Cosmos (Ed25519) and Ethereum PoS (BLS12-381) signatures.

As in [LXZ+23], we use a bivariate KZG polynomial commitment scheme, which entails a universal updatable CRS linear in the circuit size. The proof size is constant, as are the verification time - dominated by three pairings - and the communication complexity between the Prover machines. With a 9-limb pairing-friendly outer curve to Ed25519, the proof size is 5 KB. With this same curve, the communication complexity for each worker node is 5 KB and that of the master node is 5 KB per machine.

The effective Prover time for a circuit of size $T \cdot M$ on M machines is $O(T \cdot \log(T) + M \cdot \log(M))$. The work of each Prover machine is dominated by the MSMs of length T in the group $\mathbb{G}_1$ and a single sum of univariate polynomial products computed via multimodular FFTs[1] of size $2T$. Likewise, the work of the master node is dominated by the MSMs of length M in the group $\mathbb{G}_1$ and a single sum of univariate polynomial products via multimodular FFTs of size $2M$.

1 Introduction

The goal of this work was to construct a Snark with the following properties/attributes:

1 Compatibility with a wider class of prime fields

In particular, we need compatibility with pairing-friendly outer curves to widely used curves such as Ed25519, secp256k1, BN254 and BLS12-381. This was the key goal of the paper.

The use cases necessitated a Snark that could sidestep the overhead of non-native field arithmetic that arises when statements in the base fields of these curves are proved using a Snark in a mismatched finite field. These use cases include EdDSA, ECDSA signatures and one layer recursion with the widely used curves BN254 and BLS12-381. As far as we know, the existing Snarks that allow for constant-sized proofs and constant verification times explicitly assume the existence of a sparse vanishing polynomial that splits completely over the scalar field and hence, need the scalar field to have a large smooth order subgroup.

2 Fully distributed proof generation with $O(1)$ communication complexity between the machines.

As in *Pianist* ([LXZ23]), we use a bivariate KZG polynomial commitment scheme, which allows for this.

[1] The Prover machines use ordinary FFTs when the 2-adicity is high enough

3 A constant-sized proof and a constant verification time.

4 A bare minimum of pairings in the verification

It is widely known that pairings are expensive, especially in curves that fall outside of highly optimized families. The pairing-friendly curves we instantiate the scheme with for our primary use cases are constructed via the Cocks-Pinch or Brezing-Weng algorithms and the pairings are not as highly optimized as those in the BLS, BN or MNT families. This makes it all the more desirable to have a bare minimum of pairings in the verification.

The verification time in our scheme is dominated by three pairings. The verication does not involve pairings with Prover defined $\mathbb{G}_2$ points,which makes recursive aggregation of proofs convenient.

5 A universal updateable trusted setup and a CRS size linear in the circuit size

We use the bivariate KZG10 commitment scheme which allows for this. While we certainly would have preferred a transparent setup, there is - as far as we know - no scheme at the moment that achieves a transparent setup in conjunction with a constant proof size, constant verification time and a quasi-linear Prover time.

6 Support for custom gates

The scheme uses Plonkish arithmetization and hence, supports custom gates, albeit at the cost of slightly larger proof sizes. Our primary use cases benefit from the use of elliptic curve custom gates since they reduce point additions and point doublings to 2 gates instead of 9.

1.1 The setup

Let $\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T$ be cyclic groups of prime order p such that there exists a *bilinear*, *non-degenerate* and *efficiently computable* pairing

$$\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$$

We fix generators $\mathbf{g}_1,\ \mathbf{g}_2$ in $\mathbb{G}_1,\ \mathbb{G}_2$ respectively. For a trapdoors $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{F}_p^*$, the common reference string (CRS) generated via a multi-party computation is given by

$$\left[\mathbf{g}_1^{\mathbf{s}_1^i \cdot \mathbf{s}_2^j}\ :\ (i,j)\ \in\ [0,\ T-1] \times [0,\ M-1]\right]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}_1},\ \mathbf{g}_2^{\mathbf{s}_2}]$$

for appropriate upper bounds N, M. The verification key is

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

We denote the KZG commitment $g_1^{f(s_1,s_2)}$ to a polynomial $f(X,Y)$ by $\left[f(X,Y)\right]_{\mathbb{G}_1}$ and the $\mathbb{G}_2$-element $g_2^{f(\mathbf{s}_1,\mathbf{s}_2)}$ by $\left[f(X,Y)\right]_{\mathbb{G}_2}$. Thus, the CRS can be rephrased as

$$\left[\left[X^iY^j\right]_{\mathbb{G}_1}\ :\ (i,j)\ \in\ [0,\ T-1] \times [0,\ M-1]\right]\ ,\ \left[\left[1\right]_{\mathbb{G}_2},\ \left[X\right]_{\mathbb{G}_2},\ \left[Y\right]_{\mathbb{G}_2}\right]$$

and the Verifier key as

$$\left[\left[1\right]_{\mathbb{G}_1},\ \left[X\right]_{\mathbb{G}_1}\ ,\ \left[X^T\right]_{\mathbb{G}_1}, \left[Y\right]_{\mathbb{G}_1}, \left[Y^M\right]_{\mathbb{G}_1}\right]\ ,\ \left[\left[1\right]_{\mathbb{G}_2},\ \left[X\right]_{\mathbb{G}_2},\ \left[Y\right]_{\mathbb{G}_2}\right]$$

1.2 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 $\mathbb{F}_q$. $\mathbb{F}_q[X,Y]$ denotes the UFD of bivariate polynomials and $\mathbb{F}_q(X,Y)$ denotes its fraction field.

For a polynomial $f(X)$, $\deg(f)$ denotes its degree. $\texttt{Coef}(f,\ i)$ denotes the coefficient at the position X^i and $\texttt{Coef}(f)$ denotes the set $\{\texttt{Coef}(f,\ i) :\ i \leq \deg(f)\}$. Similarly, for a bivariate polynomial $f(X,Y)$, $\texttt{Coef}(f,\ (i,j))$ denotes the coefficient at X^iY^j. The *Hadamard* and *dot* products of $f_1(X,Y)$ and $f_2(X,Y)$ are given by

$$f_1 \odot f_2(X,Y) := \sum_{i,j} \texttt{Coef}(f_1\ ,\ (i,j)) \cdot \texttt{Coef}(f_2\ ,\ (i,j)) \cdot X^iY^j$$

$$f_1 \circ f_2(X,Y) := \sum_{i,j} \texttt{Coef}(f_1\ ,\ (i,j)) \cdot \texttt{Coef}(f_2\ ,\ (i,j)) = f_1 \odot f_2(1,1).$$

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.

1.3 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

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

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 $\mathcal{E}_{\texttt{multi-PC}}$ 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.

1.4 Distributed KZG commitments and openings

For a polynomial

$$f(X,Y) = \sum_{j=0}^{M-1} f_j(X) \cdot Y^j,$$

we say the Prover machines *collaboratively compute* $f(X,Y)$ *and send the commitment* $\big[f(X,Y)\big]_{\mathbb{G}_1}$ as shorthand for the following process:

- Each Prover machine $\mathcal{P}_j$ computes $f_j(X)$ and sends the commitment $\big[f_j(X) \cdot Y^j\big]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$.

- The master node $\widetilde{\mathcal{P}}$ computes

$$\big[f(X,Y)\big]_{\mathbb{G}_1} \ = \ \prod_{j=0}^{M-1} \big[f_j(X) \cdot Y^j\big]_{\mathbb{G}_1}.$$

by adding the M points in $\mathbb{G}_1$.

For a committed bivariate polynomial $f(X,Y)$ and element $(\alpha,\beta) \in \mathbb{F}_p^2$ with $\theta = f(\alpha,\beta)$, we have the equation

$$f(X,Y) - \theta \ = \ (X-\alpha) \cdot \frac{f(X,Y) - f(\alpha,Y)}{X-\alpha} \ + \ (Y-\beta) \cdot \frac{f(\alpha,Y) - \theta}{Y-\beta}$$

and the opening boils down to sending the commitments

$$\mathbf{Q}_\alpha := \big[(X-\alpha)^{-1} \cdot [f(X,Y) - f(\alpha,Y)]\big]_{\mathbb{G}_1} \ , \ \mathbf{Q}_{\alpha,\beta} := \big[(Y-\alpha)^{-1} \cdot [f(\alpha,Y) - f(\alpha,\beta)]\big]_{\mathbb{G}_1}$$

which satisfy the pairing equation

$$\mathbf{e}\big(\big[f(X,Y) - \theta\big]_{\mathbb{G}_1} \ , \ \big[1\big]_{\mathbb{G}_2}\big) \ = \ \mathbf{e}\big(\mathbf{Q}_\alpha \ , \ \big[X-\alpha\big]_{\mathbb{G}_2}\big) \cdot \mathbf{e}\big(\mathbf{Q}_{\alpha,\beta} \ , \ \big[Y-\beta\big]_{\mathbb{G}_2}\big).$$

Consider a distributed setting with

$$f(X,Y) = \sum_{j=0}^{M-1} f_j(X) \cdot Y^j \ ,$$

where each $f_j(X)$ is univariate of degree $\leq T-1$ and is held by the Prover machine $\mathcal{P}_j$. Each $\mathcal{P}_j$ computes the quotient $q_j(X) := (X-\alpha)^{-1} \cdot [f_j(X) - f_j(\alpha)]$ and sends

$$f_j(\alpha) \ \ , \ \ \mathbf{Q}_{\alpha,j} := \big[q_j(X) \cdot Y^j\big]_{\mathbb{G}_1}$$

to the master node $\widetilde{\mathcal{P}}$, who then computes

$$\mathbf{Q}_\alpha \;=\; \prod_{j=0}^{M-1} \mathbf{Q}_{\alpha,j} \;\;,\;\; f(\alpha,Y) = \sum_{j=0}^{M-1} f_j(\alpha)\cdot Y^j \;\;,\;\; \mathbf{Q}_{\alpha,\beta} \;= \big[(Y-\beta)^{-1}\cdot[f(\alpha,Y)-\theta]\big]_{\mathbb{G}_1}$$

and sends $\mathbf{Q}_\alpha$, $\mathbf{Q}_{\alpha,\beta}$ to the Verifier. This entails:

- $O(T)$ work for each machine $\mathcal{P}_j$, dominated by the MSM of length T.

- $O(M)$ work for the master node $\widetilde{\mathcal{P}}$, dominated by the MSM of length M.

1.5 Commitments to index sets and permutations

For an index set $\mathcal{I} \subseteq [0,\ T-1]\times[0,\ M-1]$, we commit to $\mathcal{I}$ by committing to the polynomial

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

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

$$\mathtt{Com}(\mathcal{I}) \;:=\; \big[\chi_{\mathcal{I}}(X,Y)\big]_{\mathbb{G}_1} \;=\; \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s}_1,\mathbf{s}_2)} \;=\; \mathbf{g}_1^{\sum\limits_{(i,j)\in\mathcal{I}} {\mathbf{s}_1}^i\cdot\mathbf{s}_2^j}.$$

For a bijection $\boldsymbol{\sigma}:\ [0,\ T-1]\times[0,\ M-1] \longrightarrow\ [0,\ T\cdot M-1]$, we commit to $\boldsymbol{\sigma}$ by committing to the polynomial

$$S_{\boldsymbol{\sigma}}(X,Y) := \sum_{i=0}^{T-1}\sum_{j=0}^{M-1} \boldsymbol{\sigma}(i,j)\cdot X^iY^j.$$

In particular, we commit to the identity bijection

$$\mathbf{id}:\ [0,\ T-1]\times[0,\ M-1] \;\longrightarrow\; [0,\ T\cdot M-1] \;\;,\;\; (i,j)\mapsto i\cdot M+j$$

via the polynomial

$$S_{\mathbf{id},T,M}(X) := \sum_{i=0}^{T-1}\sum_{j=0}^{M-1} (i\cdot M+j)\cdot X^iY^j.$$

2 Building blocks

2.1 Preliminary lemmas

Let p be a prime of bitsize $\geq 2\lambda_{\mathtt{sec}}$ and d an integer ≥ 1. For brevity, we write $R := \mathbb{F}_p[X_1,\cdots,X_d]$ and denote its field of fractions $\mathbb{F}_p(X_1,\cdots,X_d)$ by $F := \mathrm{Frac}(R)$. Note that R is a UFD and in particular, is integrally closed.

Lemma 2.1. *For rational functions $h_i \in F := \mathrm{Frac}(R)$, if the sum $\sum_{i\in\mathcal{I}} \lambda^i\cdot h_i$ lies in R for a randomly generated $\lambda\in\mathbb{F}_p$, then with overwhelming probability, each rational function h_i lies in R.*

Proof. For each index $i\in\mathcal{I}$, choose elements $h_{i,1},\ h_{i,2} \in R$ such that $h_j = h_{j,1}\cdot {h_{j,2}}^{-1}$. Suppose there exists at least one index $j\in\mathcal{I}$ such that h_j does not lie in R. Then there exists a maximal ideal $\mathfrak{m}\subseteq R$ and an integer $t_j\geq 0$ such that

$$h_{j,1}\in \mathfrak{m}^{t_j}\setminus\mathfrak{m}^{t_j+1}\;,\; h_{j,2}\in\mathfrak{m}^{t_j+1}.$$

Let $R_{\mathfrak{m}}$ denote the localization of R at $\mathfrak{m}$.

Write $\widehat{h}_{k,1} := h_{k,1}^{-1} \cdot (\prod_{i \in I} h_{i,1}) \in R$ for each index $k \in \mathcal{I}$, so that

$$\sum_{i \in \mathcal{I}} \lambda^i \cdot h_i = (\prod_{i \in I} h_{i,2})^{-1} \cdot \big[\sum_{i \in \mathcal{I}} \lambda^i \cdot \widehat{h}_{i,1} \big].$$

Let $f(X) \in R(X)$ denote the univariate polynomial $\sum_{i \in \mathcal{I}} \widehat{h}_{i,1} \cdot X^i$. Let $\widetilde{F}/F$ be a splitting field of $f(X)$ and let $\widetilde{R}_{\mathfrak{m}} \subseteq \widetilde{F}$ denote the integral closure with respect to $R_{\mathfrak{m}} \subseteq F$. Fix a maximal ideal $\widetilde{\mathfrak{m}}$ in $\widetilde{R}_{\mathfrak{m}}$ lying over $\mathfrak{m}R_{\mathfrak{m}}$.

Let $\alpha_1, \cdots, \alpha_n \in \widetilde{F}$ denote the zeros of $f(X)$, so that $f(X) = c \cdot \prod_{k=1}^{n} (X - \alpha_k)$ for some $c \in F$. Now, for any element $\lambda \in \mathbb{F}_p$,

$$(\prod_{i \in I} h_{i,2})^{-1} \cdot f(\lambda) \in R \implies \lambda - \alpha_k \in \widetilde{\mathfrak{m}} \text{ for at least one index } k,$$

which occurs with probability $\leq \frac{n}{p}$. □

2.2 Batched proof of divisibility

The Snark proof generation requires verifiably sending commitments $\big[f_t(\alpha_i, Y)\big]_{\mathbb{G}_1}$ $(t = 0, \cdots, 6)$ for 7 distinct committed polynomials $f_t(X, Y)$ and elements $\alpha_t \in \mathbb{F}_p$. This requires showing that:

- the elements $\big[f_t(\alpha_t, Y)\big]_{\mathbb{G}_1}$ are commitments to univariate $\mathbb{F}_p[Y]$ polynomials, i.e. these committed polynomials have X-degrees 0

- the polynomial $h_t(X, Y) := f_t(X, Y) - f_t(\alpha_i, Y)$ is divisible by $X - \alpha_t$ for each t.

Naïvely, this would mean 7 additional $\mathbb{G}_1$-MSMs of length T for each Prover machine. Instead, we describe a protocol to show that given committed polynomials $f_t(X, Y)$ and sparse polynomials $e_t(X)$, each $f_t(X, Y)$ is divisible by $e_t(X)$, with the following costs for the Prover machines:

- Two KZG commitments to bivariate polynomials of X-degree $\leq T$ and Y-degree $\leq M$, computed in effective runtime $O(T/\log(T))$ via the distributed approach

- One KZG commitment to univariate polynomials of Y-degree $\leq M$ computed in runtime $O(M/\log(M))$ by the master node $\widetilde{\mathcal{P}}$.

The protocol hinges on the simple observation (lemma 2.1) that for a set of rational functions in $\mathbb{F}_p(X, Y) := \mathrm{Frac}(\mathbb{F}_p[X, Y])$, if a randomized sum of these rational functions is a polynomial, then with overwhelming probability, all of the rational functions are polynomials. Write $\mathbf{a}_t := \big[f_t(X, Y)\big]_{\mathbb{G}_1}$ for brevity. We impose the condition that the univariate polynomials $e_t(X, Y)$ are sparse so that the Verifier can evaluate them at a challenge. In response to a randomly generated challenge $\lambda \in \mathbb{F}_p$, the Prover machines collaboratively compute the polynomial

$$f_\lambda(X, Y) \ := \ \sum_{t=1}^{k} \lambda^{t-1} \cdot e_t(X)^{-1} \cdot f_t(X, Y)$$

and the master node sends the commitment

$$\mathbf{B}_\lambda := \big[f_\lambda(X, Y)\big]_{\mathbb{G}_1} = \prod_{j=0}^{M-1} \big[h_{\lambda,j}(X) \cdot Y^j\big]_{\mathbb{G}_1}$$

to the Verifier. Lemma 2.1 implies that proving the well-formation of $\mathbf{B}_\lambda$ is sufficient to show that the polynomial committed in the element $\mathbf{a}_t$ is divisible by $e_t(X)$ for each $t = 0, \cdots, k-1$.

Protocol 2.2. *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

$$\Big[\big[X^iY^j\big]_{\mathbb{G}_1}\ :\ (i,j)\ \in\ [0,\ T-1]\times[0,\ M-1]\Big]\ ,\ \Big[\big[1\big]_{\mathbb{G}_2},\ \big[X\big]_{\mathbb{G}_2},\ \big[Y\big]_{\mathbb{G}_2}\Big]$$

Common Inputs: Elements $\mathbf{a}_t\ \in\ \mathbb{G}_1$; sparse public polynomials $e_t(X) \in \mathbb{F}_p[X]$; $t = 0, \cdots, k$

Claim: The Prover knows polynomials $f_t(X,Y)$ such that

$$\mathbf{a}_t = \big[f_t(X,Y)\big]_{\mathbb{G}_1}\ ,\ \ f_t(X,Y) \equiv 0 \ (\mathrm{mod}\ e_t(X)).$$

Proof generation algorithm

1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widetilde{\lambda}$.
2. The Prover machines collaboratively compute the polynomial

$$f_{\widetilde{\lambda}}(X)\ :=\ \sum_{t=0}^{k}\widetilde{\lambda}^t\cdot e_t(X)^{-1}\cdot f_t(X,Y)$$

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

$$\mathbf{B}_{\widetilde{\lambda}}\ :=\ \big[f_{\widetilde{\lambda}}(X,Y)\big]_{\mathbb{G}_1}.$$

3. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates challenges $\alpha, \beta\ \in\ \mathbb{F}_p$.
4. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\theta_t := f_t(\alpha,\beta)\quad (t = 0, \cdots, k-1)$ to the Verifier $\mathcal{V}$.
5. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widetilde{\xi}$.
6. The Prover machines collaboratively compute the commitments

$$\mathbf{Q}_{\alpha,\beta} = \big[q_{\alpha,\beta}(X,Y)\big]_{\mathbb{G}_1}\ ,\ \ \mathbf{Q}_\beta := q_\beta(Y)$$

to the polynomials such that

$$(X-\alpha)\cdot q_{\alpha,\beta}(X,Y) + (Y-\beta)\cdot q_\beta(Y)\ =\ \big[f_{\widetilde{\lambda}}(X,Y) - f_{\widetilde{\lambda}}(\alpha,\beta)\big]\ +\ \sum_{t=1}^{k}\widetilde{\xi}^t\cdot[f_t(X,Y)-\theta_t]$$

and send $\mathbf{Q}_\alpha$, $\mathbf{Q}_\beta$ to the Verifier.

Verification algorithm

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

$$\widetilde{\theta}\ :=\ \big[\sum_{t=1}^{k}\widetilde{\lambda}^{t-1}\cdot\theta_t\cdot e_t(\alpha,\beta)^{-1}\big]\ +\ \sum_{t=1}^{k}\widetilde{\xi}^t\cdot\beta_t$$

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

$$(\mathbf{Q}_{\alpha,\beta})^{\mathbf{s}_1-\alpha} \cdot (\mathbf{Q}_{\beta})^{\mathbf{s}_2-\beta} \;=\; \mathbf{B}_{\widetilde{\lambda}} \cdot \big[\prod_{t=1}^{k} \mathbf{a}_t^{\widetilde{\xi}^t}\big] \cdot \mathbf{g}_1^{-\widetilde{\theta}}$$

via the pairing checks. □

2.3 The degree upper bound

Unlike Plonk ([GWC19]) and Pianist ([LXZ+23]), our scheme needs a protocol for upper bounds on the degrees of committed polynomials. We first describe how our protocol works in the univariate setting before extending the techniques to the distributed bivariate setting.

For a committed univariate polynomial $f(X)$ and an integer n, we have

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

Thus, a Prover can demonstrate this upper bound on the degree by verifiably sending the 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 for the degree upper bound 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 2.1). Thus, a Prover can demonstrate all of these degree upper bounds by verifiably sending the KZG commitment to $f_\lambda(X)$.

We now describe a degree upper bound for bivariate polynomials in a distributed setting. Consider a bivariate polynomial

$$h(X,Y) = \sum_{j=0}^{M-1} h_j(X) \cdot Y^j$$

where the univariate $h_j(X)$ is held by the Prover machine $\mathcal{P}_j$. We note that for an integer n, the following are equivalent:

- $\deg_X(h) \leq n$
- $\deg(h_j) \leq n \;\; \forall\, j \in \{0, \cdots, M-1\}$
- The univariate rational function $\widehat{h}_j(X) := X^n \cdot h_j(X^{-1})$ lies in $\mathbb{F}_p[X] \;\; \forall\, j \in \{0, \cdots, M-1\}$
- The rational function

$$\widehat{h}(X,Y) := X^n \cdot h(X^{-1}, Y) = \sum_{j=0}^{M-1} \widehat{h}_j(X) \cdot Y^j$$

lies in $\mathbb{F}_p[X,Y]$.

Each machine $\mathcal{P}_j$ computes the polynomial $\widehat{h}_j(X) := X^n \cdot h_j(X^{-1})$ and sends the commitment $\big[\widehat{h}_j(X) \cdot Y^j\big]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$, who then computes the commitment $\big[\widehat{h}(X,Y)\big]_{\mathbb{G}_1}$ by adding

M $\mathbb{G}_1$-points. It now suffices to show that for a randomly generated challenge $\alpha \in \mathbb{F}_p$, the committed polynomials $h(X,Y)$ and $\widehat{h}(X,Y)$ satisfy the equation

$$\widehat{h}(\alpha, Y) = \alpha^n \cdot h(\alpha^{-1}, Y) \ \in \ \mathbb{F}_p[Y]$$

To show that $\deg_Y(h) \leq m$, it suffices to show that the polynomial $h(\alpha, Y) \in \mathbb{F}_p[Y]$ has degree $\leq m$. To that end, each Prover machine $\mathcal{P}_j$ sends $h_j(\alpha)$ to the master node $\widetilde{\mathcal{P}}$, who in turn, verifiably sends the commitments $\big[h(\alpha,Y)\big]_{\mathbb{G}_1}$, $\big[Y^m \cdot h(\alpha, Y^{-1})\big]_{\mathbb{G}_1}$. to the Verifier.

Protocol 2.3. *Degree upper bound* (`DegUp`)

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

$$\Big[\big[X^iY^j\big]_{\mathbb{G}_1} \ : \ (i,j) \ \in \ [0,\ T-1] \times [0,\ M-1]\Big] \ , \ \Big[\big[1\big]_{\mathbb{G}_2},\ \big[X\big]_{\mathbb{G}_2},\ \big[Y\big]_{\mathbb{G}_2}\Big]$$

Common Inputs: Elements $\mathbf{a} \ \in \ \mathbb{G}_1$; integers n, m

Claim: The Prover knows a polynomial $f(X,Y)$ such that

$$\mathbf{a} = \big[f(X,Y)\big]_{\mathbb{G}_1} \ , \ \deg_X(f) \leq n \ , \ \deg_Y(f) \leq m$$

Proof generation algorithm

1. The Prover machines collaboratively compute the polynomial $\widehat{f}(X,Y) := X^n \cdot f(X^{-1}, Y)$ and send the commitment

$$\widehat{\mathbf{a}} := \big[\widehat{f}(X,Y)\big]_{\mathbb{G}_1}$$

to the Verifier.

2. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge α.

3. The Prover machines collaboratively compute the polynomial $\widehat{f}(\alpha, Y)$ and send the commitment

$$\widehat{\mathbf{a}}_\alpha := \big[\widehat{f}(\alpha,Y)\big]_{\mathbb{G}_1}$$

to the Verifier.

4. The Prover machines collaboratively compute the polynomials

$$q_\alpha(X,Y) := \frac{f(X,Y) - f(\alpha,Y)}{X - \alpha} \ , \ \widehat{q}_\alpha(X,Y) := \frac{\widehat{f}(X,Y) - \alpha^{-n} \cdot f(\alpha,Y)}{X - \alpha^{-1}}$$

and send the commitments

$$\mathbf{Q}_\alpha := \big[q_\alpha(X,Y)\big]_{\mathbb{G}_1} \ , \ \widehat{\mathbf{Q}}_\alpha := \big[\widehat{q}_\alpha(X,Y)\big]_{\mathbb{G}_1}$$

to the Verifier.

5. The master node $\widetilde{\mathcal{P}}$ computes the polynomial $f^\vee_\alpha(Y) := Y^m \cdot f(\alpha, Y^{-1})$ and sends the commitment

$$\mathbf{a}^\vee_\alpha := \big[f^\vee_\alpha(Y)\big]_{\mathbb{G}_1}$$

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

7. The master node $\widetilde{\mathcal{P}}$ sends

$$\theta := f(\alpha, \beta) \ \ , \ \ \mathbf{Q}_{\alpha,\beta} := \big[\frac{f(\alpha, Y) - \theta}{Y - \beta}\big]_{\mathbb{G}_1} \ \ , \ \ \mathbf{Q}^{\vee}_{\alpha,\beta} := \big[\frac{f^{\vee}_{\alpha}(Y) - \theta \cdot \beta^{-m}}{Y - \beta^{-1}}\big]_{\mathbb{G}_1}$$

and to the Verifier.

Verification algorithm

8. The Verifier $\mathcal{V}$ verifies the equations

$$\mathbf{Q}_{\alpha}^{\mathbf{s}_1 - \alpha} \stackrel{?}{=} \mathbf{a} \cdot \mathbf{a}_{\alpha}^{-1} \ \ , \ \ \widehat{\mathbf{Q}}_{\alpha}^{\mathbf{s}_1 - \alpha^{-1}} \stackrel{?}{=} \widehat{\mathbf{a}} \cdot \mathbf{a}_{\alpha}^{-\alpha^{-n}}$$

$$\mathbf{Q}_{\alpha,\beta}^{\mathbf{s}_2 - \beta} \stackrel{?}{=} \mathbf{a}_{\alpha} \cdot \mathbf{g}_1^{-\theta} \ \ , \ \ (\mathbf{Q}^{\vee}_{\alpha,\beta})^{\mathbf{s}_2 - \beta^{-1}} \stackrel{?}{=} \mathbf{a}^{\vee}_{\alpha} \cdot \mathbf{g}_1^{-\theta \cdot \beta^{-m}}$$

via the pairing checks.

2.3.1 Batched degree upper bounds

We note that multiple degree upper bounds can be batched. Consider k bivariate polynomials

$$f_t(X, Y) = \sum_{j=0}^{M-1} f_{t,j}(X) \cdot Y^j \quad , \quad t = 0, \cdots, k-1$$

with

$$\deg_X(f_t) = \min(\deg(f_{t,j})) \leq n_t$$

and each univariate polynomial $f_{t,j}(X)$ held by the Prover machine $\mathcal{P}_j$. We assume k is a fairly small integer. The resulting number of $\mathbb{G}_1$ elements in the proof (ergo the number of MSMs in the proof generation) will be independent of k, although the number of $\mathbb{F}_p$ elements in the proof will be $O(k)$. For the basic version of the distributed Snark with a 3-ary circuit and without custom gates or lookups, we need 10 degree upper bounds. Three of these are for the wire polynomials ($\deg_X(.) \leq T$). One is for the low degree part of the sum of twisted polynomial products ($\deg_X(.) \leq T - 1$), which is necessary for the batched Hadamard product protocol. The other six ($\deg_X(.) \leq T$) are for the polynomials arising from the LogUp-esque permutation argument.

It suffices to show that for a randomly generated challenge $\widehat{\lambda}$, the rational function given by the randomized sum

$$\widehat{f}_{\widehat{\lambda}}(X, Y) \ := \ \sum_{t=0}^{k-1} \widehat{\lambda}^t \cdot X^{n_t} \cdot f_t(X^{-1}, Y) \ = \ \sum_{j=0}^{M-1} \Big[\sum_{t=0}^{k-1} \big[\widehat{\lambda}^t \cdot X^{n_t} \cdot f_{t,j}(X^{-1})\big] \Big] \cdot Y^j$$

is a polynomial. The Prover machine $\mathcal{P}_j$ computes the polynomial

$$\widehat{f}_{\widehat{\lambda},j}(X) := \sum_{t=0}^{k-1} \widehat{\lambda}^t \cdot X^{n_t} \cdot f_{t,j}(X^{-1})$$

and sends the KZG commitment $\big[\widehat{f}_{\widehat{\lambda},j}(X) \cdot Y^j\big]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$, who, in turn, computes the KZG commitment

$$\big[\widehat{f}_{\widehat{\lambda}}(X,Y)\big]_{\mathbb{G}_1} \;=\; \prod_{j=0}^{M-1} \big[\widehat{f}_{\widehat{\lambda},j}(X)\cdot Y^j\big]_{\mathbb{G}_1}.$$

and sends it to the Verifier.

It then remains to show that that the commitment to $\widehat{f}_{\lambda}(X,Y)$ is well-formed. This can be achieved by showing that for a randomly generated challenge $(\alpha,\beta)\in\mathbb{F}_p^2$,

$$\widehat{f}_{\widehat{\lambda}}(\alpha,\beta) \;=\; \sum_{t=0}^{k-1} \widehat{\lambda}^t\cdot\alpha^{n_t}\cdot f_t(\alpha^{-1},\beta).$$

Thus, it suffices to send the evaluations $f_t(\alpha^{-1},\beta)\;\;(t=0,\cdots,k-1)$ and $f_{\widehat{\lambda}}(\alpha,\beta)$ along with a batched proof of these openings.

2.4 The Hadamard product

For bivariate polynomials $f_1(X,Y)$, $f_2(X,Y)$, the *Hadamard product* $f_1\odot f_2(X,Y)$ (alternatively denoted by $f_1(X,Y)\odot f_2(X,Y)$) is given by

$$f_1\odot f_2(X) \;:=\sum_{i,j}\texttt{Coef}(f_1\;,\;(i,j))\cdot\texttt{Coef}(f_2\;,\;(i,j))\cdot X^iY^j.$$

The *dot product* $f_1\circ f_2(X,Y)$ is the evaluation of the Hadamard product $f_1\odot f_2(X,Y)$ at $(X,Y)=(1,1)$. Note that for polynomials

$$f_t(X,Y)=\sum_{j=0}^{M-1} f_{t,j}(X)\cdot Y^j\;\;,\;\;t=1,2,$$

we have the equation

$$f_1\odot f_2(X,Y) \;=\; \sum_{j=0}^{M-1} f_{1,j}\odot f_{2,j}(X)\cdot Y^j.$$

By a *proof of a Hadamard triple* $(\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_{1,2})\in\mathbb{G}_1^3$, we refer to a succinct argument attesting to the knowledge of polynomials $f_1(X,Y)$, $f_2(X,Y)$, $f_{1,2}(X,Y)$ such that

$$\mathbf{a}_1=\big[f_1(X,Y)\big]_{\mathbb{G}_1}\;,\;\mathbf{a}_2=\big[f_2(X,Y)\big]_{\mathbb{G}_1}\;,\;\mathbf{a}_{1,2}=\big[f_{1,2}(X,Y)\big]_{\mathbb{G}_1}\;,\;f_{1,2}(X,Y)=f_1\odot f_2(X,Y).$$

We briefly review the Hadamard product protocol from [Th23] for univariate polynomials in $\mathbb{F}_p[X]$ before adapting the techniques to bivariate polynomials in $\mathbb{F}_p[X,Y]$ in a distributed setting. For univariate polynomials $f_1(X),f_2(X)$ and for a fixed integer $N\geq\deg(f_2)$ and a randomly generated challenge γ, the product

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

is a polynomial of degree

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

The product $f_{\Pi,\gamma}(X)$ can be computed via a Multimodular FFT. Its coefficient $\texttt{Coef}(f_{\Pi,\gamma}\;,\;N)$ 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.

It is straightforward to show that a committed univariate polynomial is divisible by the monomial X^{N+1}. 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 $\deg(f) \leq n$.

Lastly, for a tuple

$$(\lambda_t)_{t=0}^{k-1} \in \mathbb{F}_p^k \ , \ (\mathbf{a}_t, \ \mathbf{b}_t)_{t=0}^{k-1} \in (\mathbb{G}_1^2)^k, \ \mathbf{C} \in \mathbb{G}_1 \ \ ,$$

we use the term batched *univariate Hadamard product over* $\mathbb{F}_p[Y]$ as shorthand for the master node proving the knowledge of $a_t(Y), b_t(Y), C(Y) \in \mathbb{F}[Y]$ such that:

- $\mathbf{a}_t \ = \ \big[a_t(Y)\big]_{\mathbb{G}_1}$, $\mathbf{b}_t \ = \ \big[b_t(Y)\big]_{\mathbb{G}_1}$, $\mathbf{C} = \big[C(Y)\big]_{\mathbb{G}_1}$.
- $\sum\limits_{t=0}^{k-1} \lambda^t \cdot a_t \odot b_t(Y) = C(Y)$.

As in [Th23a], this is demonstrated by showing that for a randomly sampled $\gamma \in \mathbb{F}_p$, the polynomial $\sum\limits_{t=0}^{k-1} \lambda^t \cdot a_t(\gamma \cdot Y) \cdot Y^M \cdot b_t(Y^{-1})$ has coefficient $C(\gamma)$ at Y^M.

We now move to the distributed bivariate setting. Consider k Hadamard product equations

$$A_t \odot B_t(X,Y) = C_t(X,Y) \quad , \quad t = 0, \cdots, k-1 \tag{2.1}$$

of committed polynomials

$$A_t(X,Y) = \sum_{j=0}^{M-1} A_{t,j}(X) \cdot Y^j \ , \ B_t(X,Y) \ = \ \sum_{j=0}^{M-1} A_{t,j}(X) \cdot Y^j \ , \ C_t(X,Y) = \sum_{j=0}^{M-1} C_{t,j}(X) \cdot Y^j$$

where the polynomials $A_{t,j}(X)$, $B_{t,j}(X)$, $C_{t,j}(X)$ are held by the Prover machine $\mathcal{P}_j$.

It suffices to show that for a challenge $\lambda \in \mathbb{F}_p$ randomly and uniformly generated after the commitments $\big[A_t(X,Y)\big]_{\mathbb{G}_1}$, $\big[B_t(X,Y)\big]_{\mathbb{G}_1}$, $\big[C_t(X,Y)\big]_{\mathbb{G}_1}$ have been added to the transcript - or are linear combinations of elements added to the transcript thus far - the polynomial

$$\begin{aligned} f_{\Pi,\gamma,\lambda}(X,Y) \ &:= \ \sum_{t=0}^{k} \sum_{j=0}^{M-1} \lambda^t \cdot \big[A_{t,j}(\gamma \cdot X) \cdot X^T \cdot B_{t,j}(X^{-1})\big] \cdot Y^j \\ &= \ \sum_{j=0}^{M-1} \left[\sum_{t=0}^{k-1} \lambda^t \cdot \big[A_{t,j}(\gamma \cdot X) \cdot X^T \cdot B_{t,j}(X^{-1})\big] \right] \cdot Y^j \end{aligned}$$

is of the form

$$f_{\gamma,\lambda,-}(X,Y) \ + \big[\sum_{t=0}^{k-1} \lambda^t \cdot C_t(\gamma,Y) \ \big] \cdot X^T \ + \ X^{T+1} \cdot f_{\gamma,\lambda,+}(X,Y) \quad , \quad \deg_X(f_{\gamma,\lambda,-}) \leq T-1.$$

The X-degree upper bound on $f_{\gamma,\lambda,-}(X,Y)$ can be proven via the mechanism described in the preceding section, namely verifiably sending the commitment to the polynomial

$$\widehat{f}_{\gamma,\lambda,\alpha,-}(X,Y) := X^{T-1} \cdot f_{\gamma,\lambda,\alpha,-}(X^{-1},Y).$$

This is more optimal than sending a KZG commitment to a low degree and a high degree part for each individual Hadamard product. We note that $f_{\Pi,\gamma,\lambda}(X,Y)$ is the unique polynomial such that for any $\alpha \in \mathbb{F}_p$,

$$f_{\Pi,\gamma,\lambda}(\alpha,Y) = \sum_{t=0}^{k-1} \lambda^t \cdot [A_t(\gamma\cdot\alpha,Y)] \odot [\alpha^T \cdot B_t(\alpha^{-1},Y)]$$

and hence, verifiably committing to $f_{\Pi,\gamma,\lambda}(X,Y)$ boils down to a Hadamrd product equation over $\mathbb{F}_p[Y]$.

For a challenge α generated *after* the commitments $\left[f_{\gamma,\lambda,-}(X,Y)\right]_{\mathbb{G}_1}$, $\left[\widehat{f}_{\gamma,\lambda,\alpha,-}(X,Y)\right]_{\mathbb{G}_1}$, $\left[f_{\gamma,\lambda,+}(X,Y)\right]_{\mathbb{G}_1}$ have been sent to the Verifier, the master node needs to send the commitments to the univariate polynomials

$$f_{\gamma,\lambda,\alpha,-}(Y) \ := \ f_{\gamma,\lambda,-}(\alpha,Y) \ , \ f_{\gamma,\lambda,\alpha,+}(Y) \ := \ f_{\gamma,\lambda,+}(\alpha,Y) \ , \ C_{\gamma,\lambda}(Y) := \sum_{t=0}^{k-1} \lambda^t \cdot C_t(\gamma,Y)$$

$$A_{t,\gamma,\alpha}(Y) := A_t(\gamma\cdot\alpha,Y) \ , \ \widehat{B}_{t,\alpha}(Y) := \alpha^T \cdot B_t(\alpha^{-1},Y) \quad (t=0,\cdots,k-1).$$

Furthermore, it is necessary to prove the following claims about these committed $\mathbb{F}_p[Y]$ elements:

- $A_{t,\gamma,\alpha}(Y) \equiv A_t(X,Y) \pmod{X-\gamma\cdot\alpha} \quad (t=0,\cdots,k-1)$
- $\widehat{B}_{t,\alpha}(Y) \equiv \alpha^T \cdot B_t(X,Y) \pmod{X-\alpha^{-1}} \quad (t=0,\cdots,k-1)$
- $C_{\gamma,\lambda}(Y) \equiv \sum_{t=0}^{k-1} \lambda^t \cdot C_t(X,Y) \pmod{X-\gamma}$
- $f_{\gamma,\lambda,\alpha,-}(Y) \equiv f_{\gamma,-}(X,Y) \pmod{X-\alpha}$, $f_{\gamma,\lambda,\alpha,+}(Y) \equiv f_{\gamma,+}(\alpha,Y) \pmod{X-\alpha}$.
- $f_{\gamma,\lambda,\alpha,-}(X,Y) \equiv \alpha^{T-1} \cdot \widehat{f}_{\gamma,\lambda,\alpha,-}(X,Y) \pmod{X-\alpha^{-1}}$
- $A_{t,\gamma,\alpha}(Y)$, $\widehat{B}_{t,\alpha}(Y)$, $C_{\gamma,\lambda}(Y)$, $f_{\gamma,\lambda,\alpha,-}(Y)$, $f_{\gamma,\lambda,\alpha,+}(Y) \ \in \ \mathbb{F}_p[Y]$, i.e. the X-degrees of these committed $\mathbb{F}_p[X,Y]$-elements are 0. This can be demonstrated by showing that a randomized sum is an element of $\mathbb{F}_p[Y]$.
- The univariate Hadamard product equation

$$\sum_{t=0}^{k-1} \lambda^t \cdot A_{t,\gamma,\alpha} \odot \widehat{B}_{t,\alpha}(Y) \ = \ f_{\gamma,\lambda,\alpha,-}(Y) + \alpha^T \cdot C_{\gamma,\lambda}(Y) + \alpha^{T+1} \cdot f_{\gamma,\lambda,\alpha,+}(Y) \ \in \ \mathbb{F}_p[Y].$$

For a randomly generated challenge $\theta \in \mathbb{F}_p$, it suffices to show that the polynomial

$$\sum_{t=0}^{k-1} \lambda^t \cdot A_{t,\gamma,\alpha}(\theta\cdot Y) \cdot Y^M \cdot \widehat{B}_{t,\alpha}(Y^{-1})$$

has coefficient $f_{\gamma,\lambda,\alpha,-}(\theta) + \alpha^T \cdot C_{\gamma,\lambda}(\theta) + \alpha^{T+1} \cdot f_{\gamma,\lambda,\alpha,+}(\theta)$ at the position Y^M.

The last two claims can be proven independently by the master node $\widetilde{\mathcal{P}}$. The first four boil down to a batched divisibility subprotocol and entail two MSMs of length T in $\mathbb{G}_1$ for each worker

node $\mathcal{P}_j$. Thus, in the distributed bivariate setting, a proof of k Hadamard products of committed polynomials entails the following costs for each Prover machine $\mathcal{P}_j$:

- five MSMs of length T.

- a sum of k products over $\mathbb{F}_p[X]_{\leq T}$

Protocol 2.4. *Batched proof of Hadamard Products* (`PoHadProd`)

The CRS

$$\Big[\big[X^iY^j\big]_{\mathbb{G}_1} \ : \ (i,j) \ \in \ [0,\ T-1]\times[0,\ M-1]\Big] \ , \ \Big[\big[1\big]_{\mathbb{G}_2},\ \big[X\big]_{\mathbb{G}_2},\ \big[Y\big]_{\mathbb{G}_2}\Big]$$

Common Inputs: Elements $\mathbf{A}_t,\ \mathbf{B}_t,\ \mathbf{C}_t \ \in \ \mathbb{G}_1 \quad (t=0\cdots,k-1)$

Claim: The Prover knows polynomials $A_t(X,Y)$, $B_t(X,Y)$ such that:

$$\big[A_t(X,Y)\big]_{\mathbb{G}_1}=\mathbf{A}_t \ , \ \big[B_t(X,Y)\big]_{\mathbb{G}_1}=\mathbf{B}_t \ , \ \big[A_t\odot B_t(X,Y)\big]_{\mathbb{G}_1}=\mathbf{C}_t$$

Proof generation

Randomized sum of twisted products

1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates challenges $\gamma,\ \lambda$.

2. Each worker node $\mathcal{P}_j$ sends the element $C_{t,j}(\gamma)$ to the master node $\widetilde{\mathcal{P}}$.

3. The master node $\widetilde{\mathcal{P}}$ computes the $\mathbb{F}_p[Y]$ polynomial

$$C_{\gamma,\lambda}(Y):=\sum_t \lambda^t\cdot C_t(\gamma,Y)=\sum_{t=0}^{k-1}\sum_{j=0}^{M-1}\lambda^t\cdot C_t(\gamma)\cdot Y^j$$

4. Each worker node $\mathcal{P}_j$ computes the polynomial

$$f_{\gamma,\lambda,j}(X) \ := \ \sum_{t=1}^{k}\lambda^{t-1}\cdot A_{t,j}(\gamma\cdot X)\cdot X^T\cdot B_{t,j}(X^{-1})$$

The low degree part

5. Each worker node $\mathcal{P}_j$ computes the residue

$$f_{\gamma,\lambda,-,j}(X):=\ f_{\gamma,\lambda}(X)\ (\mathrm{mod}\ X^T)$$

and sends the $\mathbb{G}_1$-element $\mathbf{a}_{-,j} \ := \ \big[f_{\gamma,\lambda,-}(X)\cdot Y^j\big]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$.

Degree upper bound on the low degree part

6. Each worker node $\mathcal{P}_j$ computes the polynomial

$$\widehat{f}_{\gamma,\lambda,-,j}(X) := X^{T-1} \cdot f_{\gamma,\lambda,-}(X^{-1})$$

and sends the $\mathbb{G}_1$-element $\widehat{\mathbf{a}}_{-,j} := \left[\widehat{f}_{\gamma,\lambda,-}(X) \cdot Y^j\right]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$.

The high degree part

7. Each worker node $\mathcal{P}_j$ computes the polynomial

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

and sends the $\mathbb{G}_1$-element $\mathbf{a}_{+,j} := \left[f_{\gamma,\lambda,+,j}(X) \cdot Y^j\right]_{\mathbb{G}_1}$ to the master node $\widetilde{\mathcal{P}}$.

8. The master node $\widetilde{\mathcal{P}}$ computes the $\mathbb{G}_1$-elements

$$\mathbf{a}_- := \prod_{j=0}^{M-1} \mathbf{a}_{-,j}\ \ ,\ \ \widehat{\mathbf{a}}_- := \prod_{j=0}^{M-1} \widehat{\mathbf{a}}_{-,j}\ \ ,\ \ \mathbf{a}_+ := \prod_{j=0}^{M-1} \mathbf{a}_{+,j}$$

and sends them to the Verifier.

The evaluation challenge

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

10. Each worker node $\mathcal{P}_j$ sends the $\mathbb{F}_p$-elements

$$A_{t,j}(\gamma \cdot \alpha)\ ,\ B_{t,j}(\alpha^{-1}) \qquad (t = 0, \cdots, k-1)$$

$$\sum_{t=0}^{k-1} \lambda^t \cdot A_{t,j} \odot B_{t,j}(\gamma)\ \ ,\ \ f_{\gamma,\lambda,-,j}(\alpha)\ \ ,\ \ f_{\gamma,\lambda,+,j}(\alpha)$$

to the master node $\widetilde{\mathcal{P}}$.

11. The master node $\widetilde{\mathcal{P}}$ computes the $\mathbb{F}_p[Y]$-elements

$$A_t(\gamma \cdot \alpha, Y)\ \ ,\ \ \alpha^T \cdot B_t(\alpha^{-1}, Y)\ \ ,\ \ C_{\gamma,\lambda}(Y) := \sum_{t=0}^{k-1} \lambda^t \cdot A_t \odot B_t(\gamma, Y)$$

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

$$\mathbf{A}_{t,\gamma,\alpha,Y} := \left[A_t(\gamma \cdot \alpha, Y)\right]_{\mathbb{G}_1}\ \ ,\ \ \mathbf{B}_{t,\alpha,Y} := \left[\alpha^T \cdot B_t(\alpha^{-1}, Y)\right]_{\mathbb{G}_1}\ \ ,\ \ \mathbf{C}_{\alpha,\lambda,Y} := \left[C_{\gamma,\lambda}\right]_{\mathbb{G}_1}$$

$$\mathbf{a}_{-,Y} := \left[f_{\gamma,\lambda,-}(\alpha, Y)\right]_{\mathbb{G}_1}\ \ ,\ \ \left[\mathbf{a}_{+,Y} := f_{\gamma,\lambda,+}(\alpha, Y)\right]_{\mathbb{G}_1}$$

to the Verifier.

12. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates challenge λ_1, β.

The batched divisibility subprotocol

13. The Prover machines send a batched divisibility proof for the following $(\mathbb{G}_1, \mathbb{F}_p[X]_{\leq 1})$ pairs:

- $\prod_{t=0}^{k-1} (\mathbf{A}_t \cdot \mathbf{A}_{t,\gamma,\alpha,Y}^{-1})^{\lambda_1^t}$, $X - \gamma \cdot \alpha$

- $\prod_{t=0}^{k-1} (\mathbf{B}_t \cdot \mathbf{B}_{t,\alpha,Y}^{-\alpha^T})^{\lambda_1^t}$, $X - \alpha^{-1}$

- $\big[\prod_{t=0}^{k-1} \mathbf{C}_t^{\lambda_1^t} \big] \cdot \mathbf{C}_{\gamma,\lambda,Y}^{-1}$, $X - \gamma$

- $\mathbf{a}_- \cdot (\mathbf{a}_{-,Y})^{-1}$, $X - \alpha$

- $\mathbf{a}_+ \cdot (\mathbf{a}_{+,Y})^{-1}$, $X - \alpha$

- $\widehat{\mathbf{a}}_- \cdot \big(\mathbf{a}_{-,Y}\big)^{-\alpha^{1-T}}$, $X - \alpha^{-1}$

Master node's work over $\mathbb{F}_p[Y]$

14. The master node $\widetilde{\mathcal{P}}$ computes the polynomial

$$h(Y) := \sum_{t=0}^{k-1} \lambda_1^t \cdot A_{t,\gamma,\alpha}(Y) + \sum_{t=0}^{k-1} \lambda_1^{k+t} \cdot B_{t,\gamma,\alpha}(Y) + \lambda_1^{2k} \cdot C_{\gamma,\lambda}(Y)$$
$$+ \lambda_1^{2k+1} \cdot f_{\gamma,\lambda,-}(\alpha, Y) + \lambda_1^{2k+2} \cdot f_{\gamma,\lambda,+}(\alpha, Y)$$

and sends

$$\beta_0 := h(\beta) \ , \ \mathbf{Q}_\beta := \big[\frac{h(Y) - \beta_0}{Y - \beta} \big]_{\mathbb{G}_1}$$

to the Verifier.

15. The master node $\widetilde{\mathcal{P}}$ sends a batched Hadamard product proof over $\mathbb{F}_p[Y]$ for the tuple

$$\Big[(\lambda^t)_{t=0,\cdots,k-1} \ , \ \big(\mathbf{A}_{t,\gamma,\alpha,Y} \ , \ \mathbf{B}_{t,\alpha,Y}\big)_{t=0,\cdots,k-1} \ , \ \mathbf{a}_{-,Y} \cdot (\mathbf{C}_{\gamma,\lambda,Y})^{\alpha^T} \cdot (\mathbf{a}_{+,Y})^{\alpha^{T+1}} \Big]$$

<u>The verification</u>

16. The Verifier $\mathcal{V}$ computes the elements

$$\prod_{t=0}^{k-1} (\mathbf{A}_t \cdot \mathbf{A}_{t,\gamma,\alpha,Y}^{-1})^{\lambda_1^t} \ , \ \prod_{t=0}^{k-1} (\mathbf{B}_t \cdot \mathbf{B}_{t,\alpha,Y}^{-\alpha^T})^{\lambda_1^t} \quad \big[\prod_{t=0}^{k-1} \mathbf{C}_t^{\lambda_1^t} \big] \cdot \mathbf{C}_{\gamma,\lambda,Y}^{-1} \ ,$$

$$\mathbf{a}_{-,Y} \cdot (\mathbf{a}_{-,Y})^{-1} \ , \ \widehat{\mathbf{a}}_{-} \cdot \big(\mathbf{a}_{-,Y}\big)^{-\alpha^{1-T}} \ , \ \mathbf{a}_{+,Y} \cdot (\mathbf{a}_{+,Y})^{-1}$$

and runs the verification of the batched divisibility subprotocol.

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

$$(\mathbf{Q}_{\beta})^{\mathbf{s}_2-\beta} \stackrel{?}{=} \prod_{t=0}^{k-1} (\mathbf{A}_{t,\gamma,\alpha,Y})^{\lambda_1^t} \cdot \prod_{t=0}^{k-1} (\mathbf{B}_{t,\alpha,Y})^{\lambda_1^{t+k}} \cdot \mathbf{C}_{\gamma,\lambda,Y}^{\lambda_1^{2k}} \cdot \mathbf{a}_{-,Y}^{\lambda_1^{2k+1}} \cdot \mathbf{a}_{+,Y}^{\lambda_1^{2k+2}} \cdot \mathbf{g}_1^{-\beta_0}$$

via the pairing check

$$\mathbf{e}(\mathbf{Q}_{\beta} \ , \ [Y]_{\mathbb{G}_2}) \stackrel{?}{=} \mathbf{e}(\prod_{t=0}^{k-1} (\mathbf{A}_{t,\gamma,\alpha,Y})^{\lambda_1^t} \cdot \prod_{t=0}^{k-1} (\mathbf{B}_{t,\alpha,Y})^{\lambda_1^{t+k}} \cdot \mathbf{C}_{\gamma,\lambda,Y}^{\lambda_1^{2k}} \cdot \mathbf{a}_{-,Y}^{\lambda_1^{2k+1}} \cdot \mathbf{a}_{+,Y}^{\lambda_1^{2k+2}} \cdot \mathbf{g}_1^{-\beta_0} \cdot \mathbf{Q}_{\beta}^{\beta} \ , \ [1]_{\mathbb{G}_2}).$$

18. $\mathcal{V}$ runs the verification of the batched Hadamard product equation over $\mathbb{F}_p[Y]$. □

2.5 The permutation argument

Since the proof system uses Plonkish arithmetization, it is fundamental to have an efficient succinct permutation argument for the copy constraints. We first describe the mechanism of the permutation argument on committed bivariate polynomials in the monomial basis. We subsequently describe how the (extended) permutation argument is used to prove the copy constraints in the Snark.

Consider two committed bivariate polynomial $f(X,Y)$, $h(X,Y)$ with

$$\deg_X(f),\ \deg_X(h) \ \leq \ T-1 \ , \ \deg_Y(f),\ \deg_Y(h) \ \leq M-1.$$

Let $\widetilde{\boldsymbol{\sigma}}$ be a permutation of $[0,\ T-1] \times [0,\ M-1]$ and let

$$\boldsymbol{\sigma} : [0,\ T-1] \times [0,\ M-1] \longrightarrow [0,\ T \cdot M - 1]$$

be the composition with the map

$$[0,\ T-1] \times [0,\ M-1] \longrightarrow [0,\ T \cdot M - 1] \ , \ (i,j) \mapsto i \cdot M + j.$$

We commit to $\widetilde{\boldsymbol{\sigma}}$ (or equivalently, to $\boldsymbol{\sigma}$) via the polynomial

$$S_{\boldsymbol{\sigma}}(X,Y) := \sum_{j=0}^{M-1} \sum_{i=0}^{T-1} \boldsymbol{\sigma}(i,j) \cdot X^i Y^j.$$

As before, we commit to the identity permutation via the polynomial

$$S_{\mathbf{id}}(X,Y) = \sum_{j=0}^{M-1} \sum_{i=0}^{T-1} (i \cdot M + j) \cdot X^i Y^j.$$

We describe a protocol to show that $\boldsymbol{\sigma}$ maps $f(X,Y)$ to $h(X,Y)$ in the following sense:

$$\texttt{Coef}(f \ , \ \widetilde{\boldsymbol{\sigma}}(i,j)) \ = \ \texttt{Coef}(h \ , \ (i,j)) \ \ \forall \ \ (i,j) \in [0,\ T-1] \times [0,\ M-1].$$

Note that this is equivalent to the polynomials

$$h(X,Y) + \delta_1 \cdot S_{\boldsymbol{\sigma}}(X,Y) \ , \ \ f(X,Y) + \delta_1 \cdot S_{\mathbf{id}}(X,Y)$$

having the same multiset of coefficients for a randomly generated challenge $\delta_1 \in \mathbb{F}_p$. This, it turn, is equivalent - with overwhelming probability- to the polynomials

$$f_{\mathbf{id},\delta_1,\delta_2}(X,Y) := \; f(X) + \delta_1 \cdot S_{\mathbf{id}}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

$$h_{\boldsymbol{\sigma},\delta_1,\delta_2}(X,Y) := \; f(X,Y) + \delta_1 \cdot S_{\boldsymbol{\sigma}}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

satisfying the equation

$$\sum_{(i,j)\in[0,\,T-1]\times[0,\,M-1]} \big[\mathtt{Coef}\big(f_{\mathbf{id},\delta_1,\delta_2}\;,\;(i,j)\big)\big]^{-1} \;=\; \sum_{(i,j)\in[0,\,T-1]\times[0,\,M-1]} \big[\mathtt{Coef}\big(f_{\boldsymbol{\sigma},\delta_1,\delta_2}\;,\;(i,j)\big)\big]^{-1}$$

for randomly generated challenges $\delta_1, \delta_2 \in \mathbb{F}_p$. Thus, it suffices to show that there exist polynomials $\widehat{h}_{\boldsymbol{\sigma},\delta_1,\delta_2}(X,Y)$, $\widehat{f}_{\mathbf{id},\delta_1,\delta_2}(X,Y)$ such that:

• The X-degrees are $\leq T$ and the Y-degrees are $\leq M$.

• the following Hadamard product equations hold

$$h_{\boldsymbol{\sigma},\delta_1,\delta_2} \odot \widehat{h}_{\boldsymbol{\sigma},\delta_1,\delta_2}(X,Y) \;=\; \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1} \;=\; f_{\mathbf{id},\delta_1,\delta_2} \odot \widehat{f}_{\mathbf{id},\delta_1,\delta_2}(X,Y)$$

• The evaluations at $(X,Y)=(1,1)$ coincide, i.e.

$$\widehat{h}_{\boldsymbol{\sigma},\delta_1,\delta_2}(1,1) \;=\; \widehat{f}_{\mathbf{id},\delta_1,\delta_2}(1,1)$$

or equivalently, the difference $\widehat{h}_{\boldsymbol{\sigma},\delta_1,\delta_2}(X,Y) - \widehat{f}_{\mathbf{id},\delta_1,\delta_2}(X,Y)$ evaluates to 0 at $(1,1)$.

2.5.1 Copy constraints

We now describe how the permutation argument is incorporated into the Snark to prove the copy constraints in a distributed setting. For simplicity, we consider a two input, one output circuit with left/right/output bivariate polynomials $L(X,Y)$, $R(X,Y)$, $O(X,Y)$. We store the circuit permutation in the form of three injective maps

$$\boldsymbol{\sigma}_t :\; [0,\; T-1] \times [0,\; M-1] \;\longrightarrow\; [0,\; 3T\cdot M-1] \quad ,\;\; t=1,2,3$$

with disjoint images, i.e.

$$[0,\; 3T\cdot M-1] \;=\; \mathrm{im}(\boldsymbol{\sigma}_1) \amalg \mathrm{im}(\boldsymbol{\sigma}_2) \amalg \mathrm{im}(\boldsymbol{\sigma}_3).$$

The maps $\boldsymbol{\sigma}_t$ are committed via the preprocessed polynomials

$$S_{\boldsymbol{\sigma}_t}(X,Y) := \sum_{(i,j)\in[0,\,T-1]\times[0,\,M-1]} \boldsymbol{\sigma}_t(i,j)\cdot X^i Y^j = \sum_{j=0}^{M-1} S_{\boldsymbol{\sigma}_{t,j}}(X)\cdot Y^j$$

with the polynomials

$$S_{\boldsymbol{\sigma}_{t,j}}(X) := \sum_{i\in[0,\,M-1]} \boldsymbol{\sigma}_t(i,j)\cdot X^i \quad ,\;\; t=1,2,3$$

stored by the Prover machine $\mathcal{P}_j$. The identity bijection is stored via the maps $\mathbf{id}_t$ $(t=1,2,3)$

$$\mathbf{id}_t :\; [0,\; T-1] \times [0,\; M-1] \;\longrightarrow\; [0,\; 3T\cdot M-1] \;\;,\;\; (i,j) \mapsto (i+T\cdot(t-1))\cdot M + j$$

and the preprocessed polynomials $S_{\mathbf{id}_t}(X,Y)$ are defined accordingly. The Prover machines need to show that for randomly generated challenges δ_1, δ_2 the triplet of polynomials

$$L_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y) \;:=\; L(X,Y) + \delta_1 \cdot S_{\mathbf{id}_1}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

$$R_{\mathbf{id}_2,\delta_1,\delta_2}(X,Y) \;:=\; R(X,Y) + \delta_1 \cdot S_{\mathbf{id}_2}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

$$O_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y) \;:=\; O(X,Y) + \delta_1 \cdot S_{\mathbf{id}_3}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

have the same *sum of coefficient inverses*

$$\sum_{(i,j)\in\,[0,\,T-1]\times[0,\,M-1]} \big[\mathtt{Coef}(L_{\mathbf{id}_1,\delta_1,\delta_2},\ (i,j))\big]^{-1} + \big[\mathtt{Coef}(R_{\mathbf{id}_2,\delta_1,\delta_2},\ (i,j))\big]^{-1} + \big[\mathtt{Coef}(O_{\mathbf{id}_3,\delta_1,\delta_2},\ (i,j))\big]^{-1}$$

as the triplet

$$L_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y) \;:=\; L(X,Y) + \delta_1 \cdot S_{\boldsymbol{\sigma}_1}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

$$R_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y) \;:=\; R(X,Y) + \delta_1 \cdot S_{\boldsymbol{\sigma}_2}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

$$O_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y) \;:=\; O(X,Y) + \delta_1 \cdot S_{\boldsymbol{\sigma}_3}(X,Y) + \delta_2 \cdot \frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$$

Thus, it boils down to verifiably committing to the six unique polynomials $\widehat{L}_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y)$, $\widehat{L}_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y)$, $\widehat{R}_{\mathbf{id}_2,\delta_1,\delta_2}(X,Y)$, $\widehat{R}_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y)$, $\widehat{O}_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y)$, $\widehat{O}_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y)$ such that:

1. The X-degrees are $\leq T$ and the Y-degrees are $\leq M$
2. All six of the following Hadamard products equal $\frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}$:

$$\widehat{L}_{\mathbf{id}_1,\delta_1,\delta_2} \odot L_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y) \;\;,\;\; \widehat{L}_{\boldsymbol{\sigma}_1,\delta_1,\delta_2} \odot L_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y) \;\;,\;\; \widehat{R}_{\mathbf{id}_2,\delta_1,\delta_2} \odot R_{\mathbf{id},\delta_1,\delta_2}(X,Y)$$

$$\widehat{R}_{\boldsymbol{\sigma}_2,\delta_1,\delta_2} \odot R_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y) \;\;,\;\; \widehat{O}_{\mathbf{id}_3,\delta_1,\delta_2} \odot O_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y) \;\;,\;\; \widehat{O}_{\boldsymbol{\sigma}_3,\delta_1,\delta_2} \odot O_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y)$$

3. The two polynomials

$$\widehat{L}_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y) + \widehat{R}_{\mathbf{id}_2,\delta_1,\delta_2}(X,Y) + \widehat{O}_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y)\,,$$

$$\widehat{L}_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y) + \widehat{R}_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y) + \widehat{O}_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y)$$

have the same evaluation at $(X,Y)=(1,1)$. Equivalently, the difference evaluates to 0 at $(X,Y)=(1,1)$.

3 The batched protocol

Protocol 3.1. The distributed Snark

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.

Common preprocessed inputs:

- Integers $T,\ M$.
- The CRS $\Big[\big[X^iY^j\big]_{\mathbb{G}_1}\ :\ (i,j)\ \in\ [0,\ T]\times[0,\ M-1]\Big]\ ,\ \ \Big[\big[1\big]_{\mathbb{G}_2},\ \big[X\big]_{\mathbb{G}_2},\ \big[Y\big]_{\mathbb{G}_2}\Big]$.
- Index sets $\mathcal{I}_{\texttt{Add}},\ \mathcal{I}_{\texttt{Mul}},\ \mathcal{I}_{\texttt{Pub}} \subseteq [0,\ T-1]\times[0,\ M-1]$ corresponding to the addition, multiplication gates and public inputs respectively and their indicator polynomials $\chi_{\mathcal{I}_{\texttt{Add}}}(X,Y)$, $\chi_{\mathcal{I}_{\texttt{Mul}}}(X,Y)$, $\chi_{\mathcal{I}_{\texttt{Pub}}}(X,Y)$
- The polynomials

$$S_{\mathbf{id}_t}(X,Y) := \sum_{(i,j)\in[0,\ T-1]\times[0,\ M-1]} \big[[i+(t-1)\cdot M]\cdot T + j\big]\cdot X^iY^j \quad ,\ \ t=1,2,3.$$

- Three injective maps

$$\boldsymbol{\sigma}_t :\ [0,\ T-1]\times[0,\ M-1]\ \longrightarrow\ [0,\ 3MT-1] \quad ,\ \ t=1,2,3$$

with disjoint images, i.e.

$$[0,\ T-1]\times[0,\ 3M-1]\ =\ \mathrm{im}(\boldsymbol{\sigma}_1)\amalg\mathrm{im}(\boldsymbol{\sigma}_2)\amalg\mathrm{im}(\boldsymbol{\sigma}_3)$$

committed via the polynomials

$$S_{\boldsymbol{\sigma}_t}(X,Y) := \sum_{(i,j)\in[0,\ T-1]\times[0,\ M-1]} \boldsymbol{\sigma}_t(i,j)\cdot X^iY^j.$$

- We denote by

$$\boldsymbol{\sigma}_t : [0,\ T-1]\times[0,\ 3M-1]\ \longrightarrow\ [0,\ 3MT-1]$$

the bijection

$$(i,j+M\cdot(t-1))\ \mapsto\ \boldsymbol{\sigma}_t(i,j)\ \ \forall\ (i,j)\in[0,\ T-1]\times[0,\ M-1]\ \ ,\ \ t=1,2,3.$$

Prover Machine $\mathcal{P}_j$'s preprocessed inputs: For each polynomial

$$f(X,Y)=\sum_{j=0}^{M-1} f_j(X)\cdot Y^j$$

in the common preprocesed input, the machine $\mathcal{P}_j$ stores the polynomial $f_j(X)$.

Verifier preprocessed inputs:

$$\big[1\big]_{\mathbb{G}_1},\ \big[X\big]_{\mathbb{G}_1},\ \big[X^T\big]_{\mathbb{G}_1},\ \big[Y\big]_{\mathbb{G}_1},\ \big[Y^M\big]_{\mathbb{G}_1},\ \big[1\big]_{\mathbb{G}_2},\ \big[X\big]_{\mathbb{G}_2},\ \big[Y\big]_{\mathbb{G}_2}$$

$$\widehat{\mathbf{C}}_{\mathtt{Add}} := \big[\widehat{\chi}_{\mathcal{I}_{\mathtt{Add}}}(X,Y)\big]_{\mathbb{G}_1} \ , \ \widehat{\mathbf{C}}_{\mathtt{Mul}} := \big[\widehat{\chi}_{\mathcal{I}_{\mathtt{Mul}}}(X,Y)\big]_{\mathbb{G}_1} \ , \ \widehat{\mathbf{C}}_{\mathtt{Pub}_t} := \big[\widehat{\chi}_{\mathcal{I}_{\mathtt{Pub}_t}}(X,Y)\big]_{\mathbb{G}_1}$$

$$\mathbf{A}_{\mathtt{Pub}_t} := \big[P_{\mathtt{pub}_t}(X,Y)\big]_{\mathbb{G}_1} \ , \ \mathbf{C}_{\boldsymbol{\sigma}_t} := \big[S_{\boldsymbol{\sigma}_t}(X,Y)\big]_{\mathbb{G}_1} \ , \ \mathbf{C}_{\mathbf{id}_t} := \big[S_{\mathbf{id}_t}(X,Y)\big]_{\mathbb{G}_1}$$

$$(t = 1, 2, 3)$$

$$\mathbf{C}_1 := \big[\sum_{(i,j)\in[0,\,T-1]\times[0,\,M-1]} X^i Y^j \big]_{\mathbb{G}_1} = \big[\frac{X^T-1}{X-1} \cdot \frac{Y^M-1}{Y-1}\big]_{\mathbb{G}_1}$$

Claim: The Prover knows polynomials $L(X,Y),\ R(X,Y),\ O(X,Y)$ such that:

- $\deg_X(L),\ \deg_X(R),\ \deg_X(O) \ \leq \ T-1$
- $\deg_Y(L),\ \deg_Y(R),\ \deg_Y(O) \ \leq \ M-1$
- $[L \odot R(X,Y) - O(X,Y)] \ \odot \ \chi_{\mathcal{I}_{\mathtt{Mul}}}(X,Y) \ = \ 0$
- $[L(X,Y) + R(X,Y) - O(X,Y)] \ \odot \ \chi_{\mathcal{I}_{\mathtt{Add}}}(X,Y) \ = \ 0$
- $[L(X,Y) - P_{\mathtt{pub}_1}(X,Y)] \odot \chi_{\mathcal{I}_{\mathtt{Pub}_1}}(X,Y) \ = \ [R(X,Y) - P_{\mathtt{pub}_2}(X,Y)] \odot \chi_{\mathcal{I}_{\mathtt{Pub}_2}}(X,Y)$
  $= \ [O(X,Y) - P_{\mathtt{pub}_3}(X,Y)] \odot \chi_{\mathcal{I}_{\mathtt{Pub}_3}}(X,Y) \ = \ 0.$
- The concatenated polynomial
$$\mathtt{Wires}(X,Y) := L(X,Y) + Y^M \cdot R(X,Y) + Y^{2M} \cdot O(X,Y)$$
is stable under the action of the permutation $\boldsymbol{\sigma}$, i.e.
$$\mathtt{Coef}(\mathtt{Wires}\ ,\ \boldsymbol{\sigma}(i,j)) = \mathtt{Coef}(\mathtt{Wires}\ ,\ (i,j)) \ \ \forall\ (i,j) \in [0, T-1] \times [0, 3M-1].$$

The proof generation:

Committing to the wire polynomials

1 The Prover machines collaboratively compute the polynomials $L(X,Y)$, $R(X,Y)$, $O(X,Y)$, $L \odot R(X,Y)$ and send the commitments

$$\mathbf{A}_L \ := \ \big[L(X,Y)\big]_{\mathbb{G}_1} \ , \ \mathbf{A}_R \ := \ \big[R(X,Y)\big]_{\mathbb{G}_1}, \ \mathbf{A}_O \ := \ \big[O(X,Y)\big]_{\mathbb{G}_1} \ , \ \mathbf{A}_{L,R} \ := \ \big[L \odot R(X,Y)\big]_{\mathbb{G}_1}$$

to the Verifier $\mathcal{V}$.

The permutation polynomials

2 The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates challenges $\delta_1,\ \delta_2$.

3 The Prover machines collaboratively compute the polynomials

$$\widehat{L}_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(L\ ,\ (i,j)) + \delta_1 \cdot \mathbf{id}_1(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

$$\widehat{L}_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(L\ ,\ (i,j)) + \delta_1 \cdot \boldsymbol{\sigma}_1(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

$$\widehat{R}_{\mathbf{id}_2,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(R\ ,\ (i,j)) + \delta_1 \cdot \mathbf{id}_2(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

$$\widehat{R}_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(R\ ,\ (i,j)) + \delta_1 \cdot \boldsymbol{\sigma}_2(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

$$\widehat{O}_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(O\ ,\ (i,j)) + \delta_1 \cdot \mathbf{id}_3(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

$$\widehat{O}_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y) := \sum_{(i,j)\in[0,T-1]\times[0,M-1]} \big[\mathtt{Coef}(O\ ,\ \boldsymbol{\sigma}_3(i,j)) + \delta_1 \cdot \boldsymbol{\sigma}_3(i,j) + \delta_2\big]^{-1} \cdot X^i Y^j$$

and send the six $\mathbb{G}_1$-elements

$$\widehat{\mathbf{A}}_{L,\mathbf{id}_2,\delta_1,\delta_2} := \big[\widehat{L}_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}\ ,\ \widehat{\mathbf{A}}_{L,\boldsymbol{\sigma}_1,\delta_1,\delta_2} := \big[\widehat{L}_{\boldsymbol{\sigma}_1,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}$$

$$\widehat{\mathbf{A}}_{R,\mathbf{id}_2,\delta_1,\delta_2} := \big[\widehat{R}_{\mathbf{id}_1,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}\ ,\ \widehat{\mathbf{A}}_{R,\boldsymbol{\sigma}_2,\delta_1,\delta_2} := \big[\widehat{R}_{\boldsymbol{\sigma}_2,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}$$

$$\widehat{\mathbf{A}}_{O,\mathbf{id}_3,\delta_1,\delta_2} := \big[\widehat{O}_{\mathbf{id}_3,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}\ ,\ \widehat{\mathbf{A}}_{O,\boldsymbol{\sigma}_3,\delta_1,\delta_2} := \big[\widehat{O}_{\boldsymbol{\sigma}_3,\delta_1,\delta_2}(X,Y)\big]_{\mathbb{G}_1}$$

4 The Prover machines send a batched proof of the following Hadamard product triples of committed polynomials:

- $\mathbf{A}_L\ ,\ \mathbf{A}_R\ ,\ \mathbf{A}_{L\odot R}$
- $\mathbf{A}_L \cdot \mathbf{A}_R \cdot \mathbf{A}_O^{-1}\ ,\ \mathbf{C}_{\mathtt{add}}\ ,\ 1_{\mathbb{G}_1}$
- $\mathbf{A}_{L\odot R} \cdot \mathbf{A}_O^{-1}\ ,\ \mathbf{C}_{\mathtt{mul}}\ ,\ 1_{\mathbb{G}_1}$
- $\mathbf{A}_L \cdot \mathbf{C}_{\mathbf{id}_1}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{L,\mathbf{id}_1,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_L \cdot \mathbf{C}_{\boldsymbol{\sigma}_1}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{L,\boldsymbol{\sigma}_1,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_R \cdot \mathbf{C}_{\mathbf{id}_2}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{R,\mathbf{id}_2,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_R \cdot \mathbf{C}_{\boldsymbol{\sigma}_2}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{R,\boldsymbol{\sigma}_2,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_O \cdot \mathbf{C}_{\mathbf{id}_3}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{O,\mathbf{id}_3,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_O \cdot \mathbf{C}_{\boldsymbol{\sigma}_3}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ ,\ \widehat{\mathbf{A}}_{O,\boldsymbol{\sigma}_3,\delta_1,\delta_2}\ ,\ \mathbf{C}_1$
- $\mathbf{A}_L \cdot \mathbf{A}_{\mathtt{Pub}_1}^{-1}\ ,\ \mathbf{C}_{\mathtt{Pub}_1}\ ,\ 1_{\mathbb{G}_1}$
- $\mathbf{A}_R \cdot \mathbf{A}_{\mathtt{Pub}_2}^{-1}\ ,\ \mathbf{C}_{\mathtt{Pub}_2}\ ,\ 1_{\mathbb{G}_1}$
- $\mathbf{A}_O \cdot \mathbf{A}_{\mathtt{Pub}_3}^{-1}\ ,\ \mathbf{C}_{\mathtt{Pub}_3}\ ,\ 1_{\mathbb{G}_1}$

5 The Prover machines collaboratively send a degree upper bound proof to show that the polynomials committed in

$$\mathbf{A}_L\ ,\ \mathbf{A}_R\ ,\ \mathbf{A}_O\ ,$$

$$\widehat{\mathbf{A}}_{L,\mathbf{id}_1,\delta_1,\delta_2}\ ,\ \widehat{\mathbf{A}}_{R,\mathbf{id}_2,\delta_1,\delta_2}\ ,\ \widehat{\mathbf{A}}_{O,\mathbf{id}_3,\delta_1,\delta_2}\ ,\ \widehat{\mathbf{A}}_{L,\boldsymbol{\sigma}_1,\delta_1,\delta_2}\ ,\ \widehat{\mathbf{A}}_{R,\boldsymbol{\sigma}_2,\delta_1,\delta_2}\ ,\ \widehat{\mathbf{A}}_{O,\boldsymbol{\sigma}_3,\delta_1,\delta_2}$$

have X-degrees $\leq T$ and Y-degrees $\leq M$.

6 The Prover machines collaboratively send an opening proof to show that the polynomial committed in the element

$$\widehat{\mathbf{A}}_{L,\mathbf{id}_1,\delta_1,\delta_2} \cdot \widehat{\mathbf{A}}_{R,\mathbf{id}_2,\delta_1,\delta_2} \cdot \widehat{\mathbf{A}}_{O,\mathbf{id}_3,\delta_1,\delta_2} \cdot \left[\widehat{\mathbf{A}}_{L,\boldsymbol{\sigma}_1,\delta_1,\delta_2} \cdot \widehat{\mathbf{A}}_{R,\boldsymbol{\sigma}_2,\delta_1,\delta_2} \cdot \widehat{\mathbf{A}}_{O,\boldsymbol{\sigma}_3,\delta_1,\delta_2}\right]^{-1}$$

evaluates to 0 at $(X, Y) = (1, 1)$.

The verification:

7 The Verifier $\mathcal{V}$ verifies the degree upper bounds, the batched Hadamard product protocol and the opening proof from Step 6.

Each worker node's work is dominated by the 15 MSMs of length T and to a lesser extent, the single sum of polynomial products computed via multimodular FFTs of size $2T$.

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

[EG23] 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

[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.

[LXZ+23] Liu et al, *Pianist: Scalable zkRollups via Fully Distributed Zero-Knowledge Proofs*, https://eprint.iacr.org/2023/1271

[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 Multimodular FFTs

A consequence of using the monomial basis is that the only superlinear computations the Prover performs are products of polynomials in $\mathbb{F}_p[X]$. More precisely, the only superlinear computation is the sum of twisted polynomial products arising from the bathced Hadamard product equation. 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 polynomials $f_1(X),\ f_2(X)$ to characteristic zero. Denote these polynomials by $\widetilde{f_1}(X),\ \widetilde{f_2}(X)\ \in\ \mathbb{Z}^{\geq 0}[X]$

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

3. Use the Chinese remainder theorem on the coefficients of these polynomials to recover the polynomial $\widetilde{f}_{1,2}(X) := \widetilde{f_1}(X) \cdot \widetilde{f_2}(X) \in \mathbb{Z}^{\geq 0}[X]$.

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.

B Deferred proofs

Proposition B.1. *The protocol* `BatchDiv` *is secure in the algebraic group model.*

Proof. (Sketch) The completeness is straightforward. It suffices to prove soundness. Suppose a PPT algorithm $\mathcal{A}_{\texttt{PPT}}$ outputs an accepting transcript consisting of the $\mathbb{G}_1$ elements $\mathbf{B}_{\widetilde{\lambda}}$, $\mathbf{Q}_{\alpha,\beta}$, $\mathbf{Q}_{\beta}$ and the $\mathbb{F}_p$-elements $\theta_0, \cdots, \theta_{k-1}$.

Since the challenges $\alpha,\ \beta$ were randomly and uniformly generated after the elements $\mathbf{B}_{\widetilde{\lambda}}$ and $\theta_0, \cdots, \theta_{k-1}$ were sent, the pairings checks imply that with overwhelming probability, an extractor $\mathcal{E}_{\texttt{PPT}}$ can run the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to extract a polynomial $f^*(X, Y)$ such that

$$\mathbf{B}_{\widetilde{\lambda}} \cdot \big[\prod_{t=0}^{k-1} (\mathbf{a}_t)^{\widetilde{\xi}^t} \big] \ = \ \big[f^*(X,Y) \big]_{\mathbb{G}_1} \ , \ \ f^*(\alpha,\beta) = \widetilde{\theta}.$$

The Verifier independently computes the $\mathbb{F}_p$-element

$$\widetilde{\theta} \ := \ \big[\sum_{t=0}^{k-1} \widetilde{\lambda}^{t-1} \cdot \theta_t \cdot e_t(\alpha)^{-1} \big] + \sum_{i=1}^{k} \widetilde{\xi}^i \cdot \beta_i.$$

Hence, it follows that with overwhelming probability, the extracted polynomial $f^*(X,Y)$ satisfies the equation

$$f^*(\alpha,\beta) = [\sum_{t=o}^{k-1} \widetilde{\lambda}^t \cdot \beta_t \cdot e_t(\alpha)^{-1}] + \sum_{i=1}^{k} \widetilde{\xi}^i \cdot \beta_i.$$

Since the challenge $\widetilde{\xi}$ was randomly and uniformly generated after the $\mathbb{G}_1$ element $\mathbf{B}_{\widetilde{\lambda}}$ and the $\mathbb{F}_p$-elements $\beta_1, \cdots, \beta_k$ were sent, it follows that with overwhelming probability, $\mathcal{E}_{\texttt{PPT}}$ can run the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to extract polynomials $f^*_\lambda(X,Y)$ and $f^*_t(X,Y)$ $(t = 0, \cdots, k-1)$ such that

$$\mathbf{B}_{\widetilde{\lambda}} = \big[f^*_\lambda(X,Y) \big]_{\mathbb{G}_1} \ , \ \ f^*_\lambda(\alpha,\beta) = \widetilde{\theta} \ , \ \ \mathbf{a}_t = \big[f^*_t(X,Y) \big]_{\mathbb{G}_1} \ , \ \ f^*_t(\alpha,\beta) = \theta_t \ \ (t = 0, \cdots, k-1).$$

Thus, with overwhelming probability, the extracted polynomials $f^*_\lambda(X,Y)$ and $f^*_t(X,Y)$ $(t = 0, \cdots, k-1)$ satisfy the equation

$$f^*_\lambda(\alpha,\beta) \ = \ \big[\sum_{t=0}^{k-1} \widetilde{\lambda}^t \cdot \theta_t \cdot e_t(\alpha)^{-1} \big] \ + \ \sum_{t=0}^{k-1} \widetilde{\xi}^t \cdot \beta_i \ = \ \big[\sum_{t=0}^{k-1} \widetilde{\lambda}^t \cdot \beta_t \cdot e_t(\alpha)^{-1} \big] \ + \ \sum_{t=0}^{k-1} \widetilde{\xi}^t \cdot f^*_t(\alpha,\beta).$$

Since the challenges α, β were randomly and uniformly generated after the element $\mathbf{B}_{\widetilde{\lambda}}$ was sent, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomials $f^*_\lambda(X,Y)$, $f^*_t(X,Y)$ $(t = 0, \cdots, k-1)$ satisfy the equation

$$f^*_\lambda(X,Y) \ = \ \big[\sum_{t=0}^{k-1} \widetilde{\lambda}^t \cdot f^*_t(X) \cdot e_t(X)^{-1} \big] \ + \ \sum_{t=0}^{k-1} \widetilde{\xi}^t \cdot f^*_t(X).$$

In particular, it follows that with overwhelming probability, the rational function

$$\sum_{t=0}^{k-1} \widetilde{\lambda}^t \cdot f^*_t(X) \cdot e_t(X)^{-1} \ \in \ \mathbb{F}_p(X)$$

formed by the extracted polynomials $f^*_t(X,Y)$ and the public polynomials $e_i(X)$ is a polynomial in $\mathbb{F}_p[X]$. The challenge $\widetilde{\lambda}$ was randomly and uniformly generated and hence, lemma 2.1 implies that with overwhelming probability, each of the rational functions $f^*_t(X,Y) \cdot e_j(X)^{-1}$ is a polynomial, whence it follows that $e_t(X)$ divides $f^*_t(X,Y)$ for each index $t = 0, \cdots, k-1$. □

Proposition B.2. *The protocol* `HadProd` *is secure in the algebraic group model.*

Proof. (Sketch) The completeness is straightforward. It suffices to prove soundness. Suppose a PPT algorithm $\mathcal{A}_{\texttt{PPT}}$ outputs an accepting transcript. The equation

$$(\mathbf{Q}_\beta)^{\mathbf{s}_2 - \beta} \stackrel{?}{=} \prod_{t=0}^{k-1} (\mathbf{A}_{t,\gamma,\alpha,Y})^{\lambda_1^t} \cdot \prod_{t=0}^{k-1} (\mathbf{B}_{t,\alpha,Y})^{\lambda_1^{t+k}} \cdot \mathbf{C}_{\gamma,\lambda,Y}^{\lambda_1^{2k}} \cdot \mathbf{a}_{-,Y}^{\lambda_1^{2k+1}} \cdot \mathbf{a}_{+,Y}^{\lambda_1^{2k+2}} \cdot \mathbf{g}_1^{-\beta_0}$$

implies that with overwhelming probability, an extractor $\mathcal{E}$ can run the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to obtain a polynomial $h^*(Y) \in \mathbb{F}_p[Y]$ such that

$$\big[h^*(Y)\big]_{\mathbb{G}_1} = \prod_{t=0}^{k-1} (\mathbf{A}_{t,\gamma,\alpha,Y})^{\lambda_1^t} \cdot \prod_{t=0}^{k-1} (\mathbf{B}_{t,\alpha,Y})^{\lambda_1^{t+k}} \cdot \mathbf{C}_{\gamma,\lambda,Y}^{\lambda_1^{2k}} \cdot \mathbf{a}_{-,Y}^{\lambda_1^{2k+1}} \cdot \mathbf{a}_{+,Y}^{\lambda_1^{2k+2}} \quad , \quad h(\beta) = \beta_0.$$

Since the challenge λ_1 was randomly and uniformly generated, it follows that with overwhelming probability, $\mathcal{E}$ can run the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to obtain polynomials $A_t^*(Y)$, $B_t^*(Y)$, $C_\lambda^*(Y)$, $f_{\gamma,\lambda,-,Y}(Y)$, $f_{\gamma,\lambda,+,Y}(Y)$ such that

$$\big[A^*_{t,\gamma,\alpha,Y}(Y)\big]_{\mathbb{G}_1} = \mathbf{A}_{t,\gamma,\alpha,Y} \quad , \quad \big[\widehat{B}^*_{t,\alpha,Y}(Y)\big]_{\mathbb{G}_1} = \mathbf{B}_{t,\alpha,Y} \quad , \quad \big[C^*_{\gamma,\lambda,Y}(Y)\big]_{\mathbb{G}_1} = \mathbf{C}_{\gamma,\lambda,Y}$$

$$\big[f^*_{\gamma,\lambda,-,Y}(Y)\big]_{\mathbb{G}_1} = \mathbf{a}_{-,Y} \quad , \quad \big[f^*_{\gamma,\lambda,+,Y}(Y)\big]_{\mathbb{G}_1} = \mathbf{a}_{+,Y}$$

Furthermore, $\mathcal{E}$ can run the extractor of the univariate Hadamard product protocol ([Th23a]) over $\mathbb{F}_p[Y]$. Hence, it follows that with overwhelming probability, these extracted $\mathbb{F}_p[Y]$ elements satisfy the equation

$$\sum_{t=0}^{k-1} \lambda^t \cdot A^*_{t,\gamma,\alpha,Y} \odot \widehat{B}^*_{t,\alpha,Y}(Y) \ == f^*_{\gamma,\lambda,-,Y}(Y) + \alpha^T \cdot C^*_{\gamma,\lambda,Y}(Y) + \alpha^{T+1} \cdot f^*_{\gamma,\lambda,+,Y}(Y)$$

Now, $\mathcal{E}$ can, with overwhelming probability, run the extractor of the batched divisibility subprotocol to obtain polynomials $A_t^*(X,Y)$, $B_t^*(X,Y)$, $C_t^*(X,Y)$, $f_-^*(X,Y)$, $\widehat{f}_-^*(X,Y)$, $f_+^*(X,Y)$ such that

$$\big[A^*_t(X,Y)\big]_{\mathbb{G}_1} = \mathbf{A}_t \quad , \quad \big[B^*_t(X,Y)\big]_{\mathbb{G}_1} = \mathbf{B}_t \quad , \quad \big[C^*_t(X,Y)\big]_{\mathbb{G}_1} = \mathbf{C}_t$$

$$\big[f^*_-(X,Y)\big]_{\mathbb{G}_1} = \mathbf{a}_- \quad , \quad \big[\widehat{f}^*_-(X,Y)\big]_{\mathbb{G}_1} = \widehat{\mathbf{a}}_- \quad , \quad \big[f^*_+(X,Y)\big]_{\mathbb{G}_1} = \mathbf{a}_+$$

$$A^*_t(X,Y) \equiv A^*_{t,\gamma,\alpha,Y}(Y) \pmod{X - \gamma \cdot \alpha} \quad , \quad B^*_t(X,Y) \equiv \alpha^T \cdot \widehat{B}^*_{t,\gamma,\alpha,Y}(Y) \pmod{X - \alpha^{-1}}$$

$$\sum_{t=0}^{k-1} \lambda^t \cdot C^*_t(X,Y) \equiv C^*_{\gamma,\lambda,Y}(Y) \pmod{X - \gamma}$$

$$f^*_-(X,Y) \equiv f^*_{\gamma,\lambda,-,Y}(Y) \pmod{X - \alpha} \quad , \quad f^*_-(X,Y) \equiv \alpha^{1-T} \cdot \widehat{f}^*_{\gamma,\lambda,-,Y}(Y) \pmod{X - \alpha^{-1}}$$

$$f^*_+(X,Y) \equiv f^*_{\gamma,\lambda,+,Y}(Y) \pmod{X - \alpha}$$

The challenge γ was randomly and uniformly generated after the elements $\mathbf{A}_t$, $\mathbf{B}_t$, $\mathbf{C}_t$ were added to the transcript. Furthermore, the challenge α was randomly and uniformly generated after the elements $\mathbf{a}_-$, $\widehat{\mathbf{a}}_-$, $\mathbf{a}_+$ were added to the transcript. Thus, Schwartz-Zippel implies that with overwhelming probability, these extracted polynomials satisfy the equations

$$A^*_t(\gamma \cdot \alpha, Y) = A^*_{t,\gamma,\alpha,Y}(Y) \quad , \quad \alpha^T \cdot B^*_t(\alpha^{-1}, Y) = \widehat{B}^*_{t,\gamma,\alpha,Y}(Y)$$

$$\sum_{t=0}^{k-1} \lambda^t \cdot C^*_t(\gamma, Y) = C^*_{\gamma,\lambda,Y}(Y)$$

$$f^*_-(\alpha, Y) = f^*_{\gamma,\lambda,-,Y}(Y) \quad , \quad f^*_-(\alpha, Y) = \alpha^{1-T} \cdot \widehat{f}^*_{\gamma,\lambda,-,Y}(Y)$$

$$f^*_+(\alpha, Y) = f^*_{\gamma,\lambda,+,Y}(Y)$$

In particular, it follows that with overwhelming probability,

$$X^{T-1} \cdot f_-^*(X^{-1}, Y) = \widehat{f_-^*}(X, Y) \ \in \ \mathbb{F}_p[X, Y]$$

and hence, $\deg_X(f_-^*) \leq T-1$. Write

$$A_t^*(X) = \sum_{j=0}^{M-1} \cdot A_{t,j}^*(X) \cdot Y^j \ , \ A_t^*(X) = \sum_{j=0}^{M-1} \cdot B_{t,j}^*(X) \cdot Y^j \ , \ B_t^*(X) = \sum_{j=0}^{M-1} \cdot C_{t,j}^*(X) \cdot Y^j.$$

Then we have the equation

$$\sum_{t=0}^{k-1} \sum_{j=0}^{M-1} \lambda^t \cdot [A_{t,j}^*(\gamma \cdot \alpha) \cdot \alpha^T \cdot B_{t,j}^*(\alpha^{-1})] \cdot Y^j \ = \ f_-^*(\alpha, Y) + \alpha^T \cdot [\sum_{t=0}^{k-1} \lambda^t \cdot C_t^*(\gamma, Y)] + \alpha^{T+1} \cdot f_+^*(\alpha, Y)$$

and the Schwartz-Zippel lemma implies that with overwhelming probability,

$$\sum_{t=0}^{k-1} \sum_{j=0}^{M-1} \lambda^t \cdot [A_{t,j}^*(\gamma \cdot X) \cdot X^T \cdot B_{t,j}^*(X^{-1})] \cdot Y^j \ = \ f_-^*(X, Y) + X^T \cdot [\sum_{t=0}^{k-1} \lambda^t \cdot C_t^*(\gamma, Y)] + X^{T+1} \cdot f_+^*(X, Y)$$

Thus, with overwhelming probability,

$$\texttt{Coef}(A_{t,j}^*(\gamma \cdot X) \cdot X^T \cdot B_{t,j}^*(X^{-1}) \ , \ X^T) \ = \ C_t^*(\gamma) \quad \forall \ (t, j) \in [0, k-1] \times [0, M-1] \ ,$$

and the Scwartz-Zippel lemma implies with overwhelming probability,

$$A_{t,j}^* \odot B_{t,j}^*(X) \ = \ C_{t,j}^*(X) \ \ , \ \ \forall \ (t, j) \in [0, k-1] \times [0, M-1]$$

or equivalently,

$$A_t^* \odot B_t^*(X) \ = \ C_t^*(X) \quad \forall \ t \in [0, k-1].$$

□